Linux NET-3-HOWTO, Linux Nätverk.

Terry Dawson, VK2KTJ, Alessandro Rubini (maintainer), alessandro.rubini@linux.it. Svensk översättning av Tomas Carlsson, *md5tc@mdstud.chalmers.se* <mailto:md5c@mdstud.chalmers.se> v1.3, 1 April 1998. Svensk version 14 Juni 1998

Linux har ett kärnbaserat nätverksstöd som är skrivet nästan helt från noll. TCP/IP-prestandan i de senaste kärnorna gör Linux till ett värdigt alternativ även till de bästa av dess motståndare. Detta dokument beskriver hur man installerar och konfigurerar nätverksmjukvara och tillhörande verktyg i Linux.

Innehåll

1 Ändringar från version 1.2

```
Tillägg:
        Lade till information om transproxy
        Mindre fixar här och där

Rättningar/Uppdateringar:
        Ny maintainer

ToDo (ej ändrad, tyvärr):
        Lägg till traffic shaper
        Beskriv ny routing algoritm
        Lägg till kompileringsalternativ i kärnan för IPv6
        Beskriv /proc/sys/net/* .
        WanRouter enhet
```

2 Introduktion.

Den första NET-FAQen skrevs av Matt Welsh och Terry Dawson för att besvara ofta ställda frågor om nätverk för Linux, detta i en tid före Linux Documentation Project hade startat. Den täckte de tidiga utvecklingsversionerna av Linux nätverkskärna. NET-2-HOWTO tog över från NET-FAQ och var ett av de första LDP HOWTO dokumenten, det täckte version 2, och senare version 3, av Linuxkärnans nätverksmjukvara. Detta dokument tar i sin tur över från NET-2-HOWTO och behandlar endast version 3 av Linux nätverkskärna.

Tidigare versioner av detta dokument blev ganska stora beroende på den enorma mängd material som det skulle täcka. För att minska storleken har ett antal HOWTO's producerats som behandlar specifika nätverksämnen. Detta dokument kommer, där det är relevant, att tillstå med referenser till dessa och behandla de ämnen som inte täcks i andra dokument.

I April 1998 slutade Terry Dawson att uppräthålla NET-3, pga sin höga belastning. Jag är ny i jobbet, men skall göra mitt bästa för att lyckas.

2.1 Feedback

Jag uppskattar alltid feedback och särskilt värdefulla bidrag. Var vänlig att skicka alla bidrag till *email* `<mailto:rubini@linux.it>`.

2.2 Översättarens kommentarer

Först skall sägas att jag inte är någon expert på att översätta dokument från engelska till svenska. När dokumentet dessutom är fullspäckat med tekniska termer så blir det inte lättare. Jag har försökt att översätta de flesta termerna ändå och i de fall som jag inte gjort detta så bör det vara klart vad som menas. I de fall som översättningen är tveksam så har jag även nämnt den engelska termen. För övrigt så finns det många meningar som antagligen inte skulle göra något vidare intryck på en svenskalärare, men jag hoppas att det går att förstå ändå.

På ett par ställen har jag lagt till ett par meningar, jag har då markerat detta med '(SvÖ)'.

I andra sektioner av detta dokument (förutom denna) så betyder 'jag' originalförfattaren.

–Tomas Carlsson, *md5tc@mdstud.chalmers.se* `<mailto:md5c@mdstud.chalmers.se>`

3 Så här använder man detta HOWTO dokument (NET-3-HOWTO-HOWTO ?).

Formatet på detta dokument skiljer sig från tidigare versioner. Vi har grupperat om sektionerna så att det finns upplysande material i början som man kan hoppa över om man inte är intresserad, sedan kommer generellt material som man måste förstå innan man fortsätter till de teknikspecifika delarna i resten av dokumentet.

Läs de generella sektionerna

Dessa sektioner används i alla, eller nästan alla, tekniker som beskrivs senare och det är viktigt att man förstår dessa.

Se över sitt nätverk

Man bör veta hur ens nätverk är, eller kommer att vara, designat och exakt vilken hårdvara och teknik som man kommer att implementera.

Läs det teknikspecifika sektionerna som är relaterade till de krav man har

När man vet vad man vill ha så kan man ta sig an varje komponent i tur och ordning. De här sektionerna behandlar endast detaljer för en specifik teknik.

Gör konfigurationsarbetet

Man bör försöka att konfigurera sitt nätverk och noggrant notera alla problem man stöter på.

Leta efter ytterligare hjälp om det behövs

Om man stöter på problem som detta dokument inte kan hjälpa till med så läs sektionen om var man kan hitta hjälp eller var man kan rapportera buggar.

Ha kul!

Nätverk är roligt, njut av det.

4 Generell Information om Linux Nätverk.

4.1 En kort historia om utvecklingen av Linux Nätverkskärna.

Att utveckla en helt ny implementering av TCP/IP protokollstacken i kärnan som skulle prestera lika bra som existerande implementeringar var inte en lätt uppgift. Valet att inte porta en av de existerande implementeringarna gjordes vid en tid då det rådde osäkerhet beträffande om de existerande implementeringarna kunde bli besvärade av restriktiva copyrights på grund av ett rättsfall av U.S.L och då det fanns mycket entusiasm för att göra det annorlunda och kanske till och med bättre än det redan hade gjorts.

Den första frivilliga att leda utvecklingen av nätverkskoden var Ross Biro <biro@yggdrasil.com>. Ross gjorde en enkel och inkomplett men mestadels användbar implementering, som utökades med en Ethernet-drivrutin för ett WD-8003 nätverkskort. Detta räckte för att få många personer att testa och experimentera med koden och några klarade till och med av att ansluta maskiner till Internet med denna implementering. Trycket på nätverksstöd ökade i Linuxvärlden och till slut var kostnaden för orättvist tryck på Ross större än vad han fick ut av projektet så han lade av som huvudutvecklare. Ross ansträngningar för att få igång projektet och ta ansvar för att verkligen producera någonting användbart under sådana kontroversiella omständigheter var vad som drev på allt framtida arbete och var därför en viktig del av framgången med den nuvarande produkten.

Orest Zborowski <obz@Kodak.COM> producerade det första programmeringsgränssnittet för BSD sockets till Linuxkärnan. Detta var ett stort steg framåt eftersom det tillät många av de existerande nätverksapplikationerna att bli portade till Linux utan omfattande modifiering.

Någon gång vid denna tid utvecklade Laurence Culhane <loz@holmes.demon.co.uk> de första drivrutinerna för Linux som stödde SLIP-protokollet. Dessa gjorde det möjligt för många människor som inte hade tillgång till Ethernet-nätverk att experimentera med den nya nätverksmjukvaran. Återigen så tog några personer denna drivrutin och lyckades ansluta sig till Internet. Detta gav många en smak av de möjligheter som kunde bli realiserade om Linux hade fullt nätverksstöd och ökade antalet användare av den existerande nätverksmjukvaran.

En av de som aktivt hade arbetat med uppgiften att implementera nätverksstöd var Fred van Kempen <waltje@uwalt.nl.mugnet.org>. Efter en tid av osäkerhet efter Ross avhopp från positionen som huvudutvecklare så erbjöd Fred sin tid och insats. Fred hade några ambitiösa planer för vart han ville leda Linux nätverksmjukvara och han började framskrida i den riktningen. Fred producerade en serie nätverkskod som kallades 'NET-2' kärnkod ('NET'-koden var Ross) som många kunde använda ganska användbart. Fred's NET-2 kod användes av ett hyfsat stort antal entusiaster, som ökade allt eftersom ryktet spred sig att koden fungerade. Vid denna tid var nätverksmjukvaran fortfarande ett stort antal patchar till standardversionen av kärnan och var inte inkluderad i den normala versionen. NET-FAQ och NET-2-HOWTO beskrev då den ganska komplicerade rutinen för att få allt att fungera. Freds mål var att utveckla nyheter till standard implementeringen, och detta tog tid. Linuxvärlden väntade otåligt på något som fungerade pålitligt och, precis som med Ross, så ökade trycket på Fred som huvudutvecklare.

Alan Cox <iialan@www.uk.linux.org> föreslog en lösning till problemet. Han föreslog att han skulle ta Freds NET-2 kod och debugga den, göra den pålitlig och stabil så att den skulle tillfredsställa de otåliga användarna och samtidigt lätta på trycket på Fred så att han kunde fortsätta sitt jobb. Alan började med detta, med viss framgång och hans första version av nätverkskod kallades 'NET-2D(ebugged)'. Koden fungerade pålitligt i många typiska konfigurationer och användarna var nöjda. Alan hade egna ideer och färdigheter att bidra med till projektet vilket ledde till många diskussioner relaterade till riktningen på NET-2 koden. Det skapades två olika läger i Linuxvärlden, en med filosofin 'få det att fungera, sen gör det bättre' och den andra 'gör det bättre först'. Linus stödde Alans utveckling och inkluderade Alans kod i standarddistributionen av kärnkoden. Detta placerade Fred i en svår situation. Fortsatt utveckling skulle sakna användning och testning av den stora användarskaran och detta skulle betyda att det skulle bli svårt och gå långsamt. Fred fortsatte att arbeta en kort tid men lade till slut av och Alan blev den nya ledaren för Linux nätverksutveckling.

Snart avslöjade Donald Becker <becker@cesdis.gsfc.nasa.gov> sina talanger i lågnivåaspekterna för nätverk och producerade en enorm mängd av Ethernet-drivrutiner, nästan alla de som finns i nuvarande kärnor är utvecklade av Donald. Det finns andra som har givit betydelsefulla bidrag, men Donalds arbete är produktivt och kräver speciell uppmärksamhet.

Alan fortsatte att förfina NET-2-Debugged koden en tid samtidigt som han arbetade på det som fanns på 'TODO'-listan. Då Linux `1.3.*` kärnkoden hade växt till sig hade nätverkskoden gått över till NET-3 versionen som nuvarande versioner grundar sig på. Alan arbetade på många olika aspekter på nätverkskoden och med hjälp av en mängd talangfulla personer från Linuxvärlden växte koden i alla möjliga riktningar. Alan producerade dynamiska nätverksenheter och de första standard AX.25 och IPX implementeringarna. Alan har fortsatt att meka med koden, långsamt strukturerat om och förbättrat den till det tillstånd den är i idag.

PPP-stöd lades till av Michael Callahan <callahan@maths.ox.ac.uk> och Al Longyear <longyear@netcom.com>, detta var också kritiskt för att öka antalet personer som använde Linux för nätverk.

Jonathon Naylor <jsn@cs.nott.ac.uk> har bidragit med att göra betydande förbättringar av Alans AX.25 kod, lagt till stöd för NetRom och Rose protokoll. AX.25/NetRom/Rose stödet i sig själv är ganska betydelsefullt, för att inget annat operativsystem har inbyggt stöd för dessa protokoll förutom Linux.

Det finns naturligtvis hundratals andra personer som har givit betydelsefulla bidrag till utvecklingen av Linux nätverksmjukvara. Några av dessa kommer att nämnas senare i de teknikspecifika sektionerna, andra har bidragit med moduler, drivrutiner, buggfixar, förslag, testrapporter och moraliskt stöd. I alla dessa fall kan var och en säga att den varit del av utvecklingen. Linux nätverkskod är ett utmärkt exempel på de resultat som kan uppnås med Linux anarkistiska utvecklingsstil, om man inte är förvånad över detta än, så blir man det snart för utvecklingen har inte slutat än.

4.2 Var man hittar ytterligare information om Linux nätverk.

Det finns ett antal ställen där man kan hitta bra information om Linux nätverk.

Alan Cox, den nuvarande handhavaren av Linux nätrverkskod har en World Wide Web sida som innehåller höjdpunkter från utvecklingen av nätverkskoden: *www.uk.linux.org* `<http://www.uk.linux.org/NetNews.html>`.

Ett annat bra ställe är en bok, `Network Administrators Guide`, skriven av Olaf Kirch. Den tillhör *Linux Documentatation Project* `<http://sunsite.unc.edu/LDP/>` och man kan läsa den online på *Network Administrators Guide HTML version* `<http://sunsite.unc.edu/LDP/LDP/nag/nag.html>` eller hämta den i olika format via ftp från *sunsite.unc.edu LDP ftp archive* `<ftp://sunsite.unc.edu/pub/Linux/docs/LDP/network-guide/>`. Olafs bok är ganska omfattande och ger en bra översikt av högnivåkonfiguration av nätverket i Linux.

Det finns en nyhetsgrupp i Linux nyhetshierarkin som är avsedd för nätverk och dess relaterade ämnen, den är: *comp.os.linux.networking* `<news:comp.os.linux.networking>`

Det finns en mailing-lista som man kan prenumerera på där man kan fråga frågor relaterade till Linux nätverk. För att prenumerera skall man skicka ett e-post meddelande:

```
To: majordomo@vger.rutgers.edu
Subject: anything at all
Message:

subscribe linux-net
```

På de olika IRC nätverken finns det ofta `#linux` kanaler på vilka personer kan svara på frågor om Linux nätverk.

Kom ihåg att när man rapporterar ett problem så skall man inkludera så mycket relevanta detaljer om problemet som man kan. Speciellt så bör man specificera vilka versioner av mjukvara man använder, särskilt kärnversionen, versionen på verktyg såsom *pppd* eller *dip* och den exakta karaktären av problemet som man upplever. Detta betyder bland annat att notera den exakta syntaxen på felmeddelanden som man får och alla kommandon som man använder.

4.3 Var man hittar generell nätverksinformation (icke Linux-specifik)

Om man vill ha grundläggande information om TCP/IP nätverk, så kan jag rekommendera följande dokument:

tcp/ip introduction

detta dokument finns både som *text version* `<ftp://athos.rutgers.edu/runet/tcp-ip-intro.doc>` och som *postscript version* `<ftp://athos.rutgers.edu/runet/tcp-ip-intro.ps>`.

tcp/ip administration

detta dokument finns både som *text version* `<ftp://athos.rutgers.edu/runet/tcp-ip-admin.doc>` och som *postscript version* `<ftp://athos.rutgers.edu/runet/tcp-ip-admin.ps>`.

Om man vill ha mer detaljerad information om TCP/IP nätverk så kan jag rekommendera:

```
"Internetworking with TCP/IP"
av Douglas E. Comer

ISBN 0-13-474321-0
Prentice Hall publications.
```

Man kan också prova nyhetsgruppen *comp.protocols.tcp-ip* `<news:comp.protocols.tcp-ip>`.

En viktig informationskälla när man är ute efter teknisk information relaterad till Internet och TCP/IP protokollen är RFCs. RFC är en förkortning av 'Request For Comment' och är det standardiserade sättet att lämna ut och dokumentera protokollstandarder för Internet. Det finns många ställen att hämta RFCs från. Många av dessa är ftp-sajter och andra tillhandahåller World Wide Web sidor med sökmotorer där man kan kan söka i RFC databasen med nyckelord.

En möjlig källa för RFCs är: *Nexor RFC database* `<http://pubweb.nexor.co.uk/public/rfc/index/rfc.html>`.

5 Generell Information om Nätverkskonfiguration

Följande delsektioner innehåller information som man måste förstå för att kunna konfigurera sitt nätverk. Det är fundamentala principer som gäller oavsett karaktären på nätverket som man skall använda.

5.1 Vad behöver man för att börja?

Innan man börjar bygga eller konfigurera sitt nätverk behöver man några saker. De viktigaste är:

5.1.1 Källkoden för kärnan.

Eftersom kärnan som man kör nu inte nödvändigtvis ännu har stöd för de nätverk och nätverkskort som man vill använda så behöver man antagligen källkoden för kärnan så att man kan kompilera om den med lämpliga tillägg.

Man kan alltid hämta den senaste källkoden från: *ftp.funet.fi* `<ftp://ftp.funet.fi/pub/Linux/PEOPLE/Linus/v2.0>`.

Vanligtvis så packas källkoden upp till katalogen `/usr/src/linux`. För mer information om hur man lägger till patchar och hur man kompilerar kärnan så bör man läsa *Kernel-HOWTO* `<Kernel-HOWTO.html>`. För information om hur man konfigurerar kärnans moduler så bör man läsa *Module-HOWTO* `<Module-HOWTO.html>`.

Om inte annat sägs specifikt så rekommenderar jag att man håller dig till standardversionerna av kärnan (de med jämna nummer som andra siffra i versionsnummret). Versioner under utveckling (de med udda andra siffra) kan ha strukturella eller andra ändringar som kan skapa problem med annan mjukvara i sitt system. Om man är osäker på om man klarar av att lösa den typen av problem, så skall man inte använda de versionerna.

5.1.2 Nätverksverktyg (Network Tools).

Nätverksverktygen är de program som man använder för att konfigurera nätverksenheter i Linux. Med dessa verktyg kan man tilldela adresser till enheter och konfigurera routes till exempel.

De flesta moderna Linuxdistributioner är utrustade med nätverksverktyg, så om man har installerat från en distribution och ännu inte installerat nätverksverktygen så bör man göra detta nu.

Om man inte har installerat från en distribution så behöver man källkoden för att kompilera verktygen själv. Detta är inte svårt.

Nätverksverktygen handhas av Bernd Eckenfels och finns på:*ftp.inka.de* `<ftp://ftp.inka.de/pub/comp/Linux/networking/NetTools/>` och är speglade på: *ftp.uk.linux.org* `<ftp://ftp.uk.linux.org/pub/linux/Networking/base/>`.

Man skall vara noga med att välja en version som passar den kärna som man har tänkt att använda och med att följa instruktionerna i paketet för att installera.

För att installera den senaste versionen då detta skrivs så behöver man göra följande:

```
#
# cd /usr/src
# tar xvfz net-tools-1.33.tar.gz
# cd net-tools-1.33
# make config
# make
# make install
#
```

Dessutom, om man tänker konfigurera en brandvägg eller använda IP-maskering, så behöver man *ipfwadm* kommandot. Den senaste versionen finns på: *ftp.xos.nl* `<ftp:/ftp.xos.nl/pub/linux/ipfwadm>`. Återigen så finns det ett antal versioner tillgängliga. Se till att välja den som passar bäst till din kärna.

För att installera den senaste versionen då detta skrivs så behöver man göra följande:

```
#
# cd /usr/src
# tar xvfz ipfwadm-2.3.0.tar.gz
# cd ipfwadm-2.3.0
# make
# make install
#
```

5.1.3 Applikationsprogram för nätverket

Applikationsprogrammen är program såsom *telnet* och *ftp* och deras respektive serverprogram. David Holland <dholland@hcs.harvard.edu> handhar en distribution av de vanligaste av dessa. man kan hämta dem från: *ftp.uk.linux.org* <ftp://ftp.uk.linux.org/pub/linux/Networking/base>.

För att installera den senaste versionen då detta skrivs så behöver man göra följande:

```
#
# cd /usr/src
# tar xvfz /pub/net/NetKit-B-0.08.tar.gz
# cd NetKit-B-0.08
# more README
# vi MCONFIG
# make
# make install
#
```

5.1.4 Adresser.

Internet Protokoll Adresser är uppbyggda av fyra bytes. Konventionen är att skriva adresser i vad som kallas 'punkterad decimal notation'. I denna formen är varje byte konverterad till ett decimalt tal (0-255), utan nollor i början, och skrivs med varje byte separerad med ett '.' tecken. Varje (nätverks-)gränssnitt hos en värddator eller router har en IP-adress. I vissa fall är det tillåtet att ha samma IP-adress på alla gränssnitt på en och samma maskin, men vanligtvis har varje gränssnitt sin egen adress.

Internet Protokoll nätverk är kontinuerliga sekvenser av IP-adresser. Alla adresser inom ett nätverk har ett antal siffror i adressen gemensamma. Den del av adressen som är gemensam för alla adresser inom ett nätverk kallas 'nätverksdelen' ('network portion') av adressen. Antalet bitar som är delade av alla adresser inom ett nätverk kallas 'nätmasken' ('netmask') och det är nätmaskens uppgift att avgöra vilka adresser som hör till nätverket som nätmasken tillhör och vilka som inte gör det. Beakta följande exempel:

```
-----------------  ---------------
Host Address       192.168.110.23
Network Mask       255.255.255.0
Network Portion    192.168.110.
Host portion                  .23
-----------------  ---------------
Network Address    192.168.110.0
Broadcast Address  192.168.110.255
-----------------  ---------------
```

Alla adresser som blir 'bitvis andade' med sin nätmask kommer att avslöja adressen till nätverket som den tillhör. Nätverksadressen är därför den lägst numrerade adressen i den rad av adresser som finns i nätverket och har värddatordelen av adressen satt till nollor.

Broadcastadressen är en speciell adress som varje värddator på nätverket lyssnar till utöver sin egen unika adress. Till denna adress sänds datagram om det är tänkt att alla maskiner på nätverket skall ta emot dem. Vissa typer av data såsom route information och varningsmeddelanden sänds till broadcastadressen så att alla maskiner på nätverket kan ta emot dem samtidigt. Det finns två vanligt förekommande standarder för vad broadcastadressen skall vara. Den mest använda är att den högsta möjliga adressen i nätverket används som broadcastadress. I exemplet ovan skulle detta vara `192.168.110.255`. Av någon anledning använder vissa sajter nätverksadressen som broadcastadress. I praktiken spelar det ingen större roll vilket man använder men man måste vara säker på att alla maskiner på nätverket använder samma broadcastadress.

Av administrativa skäl någon gång tidigt i utvecklingen av IP-protokollet delades adresserna in i några slumpvisa grupper av nätverk och dessa nätverk grupperades i vad som kallas för klasser. Dessa klasser tillhandahåller ett antal nätverk av standardstorlekar som kan bli allokerade. De allokerade är:

```
----------------------------------------------------------
| Network | Netmask       | Network Addresses             |
| Class   |               |                               |
----------------------------------------------------------
|    A    | 255.0.0.0     | 0.0.0.0     - 127.255.255.255 |
|    B    | 255.255.0.0   | 128.0.0.0   - 191.255.255.255 |
|    C    | 255.255.255.0 | 192.0.0.0   - 223.255.255.255 |
|Multicast| 240.0.0.0     | 224.0.0.0   - 239.255.255.255 |
----------------------------------------------------------
```

Vilka adresser som man skall använda beror på vad det är exakt man skall göra. Man kan behöva använda en kombination av följande för att få de adresser man behöver:

Installera en Linuxmaskin på ett existerande IP-nätverk

Om man vill installera en Linuxmaskin på ett existerande IP-nätverk så bör man kontakta den som administrerar nätverket och fråga efter följande:

- IP-adress för värddatorn (Host IP Address)
- IP nätverksadress (IP Network Address)
- IP broadcastadress
- IP nätmask (IP Netmask)
- Routeradress
- DNS adress (Domain Name Server Address)

Man skall sedan konfigurera sin Linux nätverksenhet med de uppgifterna. man kan inte hitta på dem och förvänta sig att det skall fungera.

Bygga ett helt nytt nätverk som aldrig skall anslutas till Internet

Om man bygger ett privat nätverk som man aldrig tänker ansluta till Internet så kan man välja vilka adresser man vill. Men, för säkerhets och konsistens skull så har vissa IP-adresser blivit reserverade speciellt för detta ändamål. De är specificerade i RFC1597 och är följande:

```
-----------------------------------------------------------
|         RESERVED PRIVATE NETWORK ALLOCATIONS            |
-----------------------------------------------------------
| Network | Netmask       | Network Addresses             |
| Class   |               |                               |
-----------------------------------------------------------
|    A    | 255.0.0.0     | 10.0.0.0    - 10.255.255.255  |
|    B    | 255.255.0.0   | 172.16.0.0  - 172.31.255.255  |
|    C    | 255.255.255.0 | 192.168.0.0 - 192.168.255.255 |
-----------------------------------------------------------
```

Man bör först bestämma hur stort nätverket skall vara och sedan välja så många adresser som man behöver.

5.2 Var skall man skriva konfigurationskommandona?

Det finns några olika metoder för Linux systemstartprocedurer. Efter det att kärnan bootar, så exekverar den alltid ett program som heter *init*. *init* programmet läser sedan en konfigurationsfil som heter `/etc/inittab` och inleder uppstartsprocessen. Det finns några olika versioner av *init* och det är denna variationen som är den största orsaken till skillnad mellan distributioner eller maskiner.

Vanligtvis så innehåller `/etc/inittab` filen en rad som ser ut någonting som liknar:

```
si::sysinit:/etc/init.d/boot
```

Denna raden specificerar namnet på det shell-script som verkligen utför uppstarten. Denna fil skulle kunna jämföras med filen `AUTOEXEC.BAT` i MS-DOS.

Vanligtvis finns det andra script som anropas av boot-scriptet och ofta så konfigureras nätverket i något av dessa.

Följande tabell kan användas som guide för ett system:

```
-------------------------------------------------------------------------------
Distrib. |Interface Config/Routing                     |Server Initialisation
-------------------------------------------------------------------------------
Debian   |/etc/init.d/network                          |/etc/init.d/netbase
         |                                             |/etc/init.d/netstd_init
         |                                             |/etc/init.d/netstd_nfs
         |                                             |/etc/init.d/netstd_misc
-------------------------------------------------------------------------------
Slackware|/etc/rc.d/rc.inet1                           |/etc/rc.d/rc.inet2
-------------------------------------------------------------------------------
RedHat   |/etc/sysconfig/network-scripts/ifup-<ifname>|/etc/rc.d/init.d/network
-------------------------------------------------------------------------------
```

De flesta moderna distributioner har ett program med vilket man kan konfigurera många av de vanligaste sorterna av nätverksgränssnitt. Om man har en av dessa så bör man se om det kan göra det man vill innan man ger sig på manuell konfiguration.

```
----------------------------------------
Distrib   | Network configuration program
----------------------------------------
RedHat    | /sbin/netcfg
Slackware | /sbin/netconfig
----------------------------------------
```

5.3 Att skapa sina nätverksgränssnitt.

I många Unix operativsystem så framträder nätverksenheterna i */dev* katalogen. Så är inte fallet i Linux. I Linux skapas nätverksenheterna dynamiskt i mjukvara och därmed måste inte enhetsfiler finnas.

I de flesta fall skapas nätverksenheterna automatiskt av drivrutinen när den initialiseras och hittar din hårdvara. Till exempel så skapar Ethernetdrivrutinen `eth[0..n]` gränssnitten sekvensiellt medan den hittar din Ethernethårdvara. Det första Ethernetkortet som hittas blir `eth0`, det andra `eth1` osv.

Men i några fall, tex *slip* och *ppp*, så skapas nätverksenheterna genom att något användarprogram körs. Samma sekvensiella numrering gäller, men enheterna skapas inte automatiskt vid systemstarten. Anledningen till detta är att, till skillnad från Ethernetenheterna, så kan antalet aktiva *slip* eller *ppp* enheter variera under tiden systemet är igång. De här fallen gås igenom mer noggrannt senare.

5.4 Att konfigurera ett nätverksgränssnitt.

När man har alla program man behöver och sina adress- och nätverksuppgifter så kan man konfigurera sina nätverksgränssnitt. När vi pratar om att konfigurera ett nätverksgränssnitt så pratar vi om processen att tilldela en nätverksenhet lämpliga adresser och att ge lämpliga värden till andra konfigurerbara parametrar hos en nätverksenhet. Det mest använda programmet för detta är kommandot *ifconfig* (interface configure).

Vanligtvis bör man använda ett kommando som liknar följande:

```
# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
```

I detta fallet konfigurerar jag ett Ethernetkort `eth0` med IP-adressen `192.168.0.1` och en nätmask `255.255.255.0`. Parametern *up* som följer kommandot talar om för enheten att den skall bli aktiv.

Kärnan förutsätter vissa förvalda värden när man konfigurerar gränssnitt. Till exempel så kan man specificera nätverksadress och broadcastadress för ett gränssnitt. Men om man inte gör det, som i exemplet ovan, så gör kärnan rimliga gissningar baserade på nätmasken, och om man inte specificerat någon nätmask så tittar kärnan på klassen av IP-adress. I exemplet ovan skulle kärnan anta att det är ett klass C nätverk som skall konfigureras och därmed sätta nätverksadressen till `192.168.0.0` och broadcastadressen till `192.168.0.255`.

Det finns många andra parametrar till *ifconfig*. De viktigaste är:

up

aktiverar ett gränssnitt.

down

deaktiverar ett gränssnitt.

[- arp]

slår av eller slår på 'address resolution protocol' för gränssnittet.

[- allmulti]

slår av eller slår på mottagning av alla hårdvaru-multicastpaket. Hårdvaru-multicast låter grupper av maskiner att ta emot paket som är adresserade till speciella destinationer. Detta kan vara viktigt om man använder applikationer såsom videokonferenssystem men används normalt inte.

mtu N

specificerar *MTU* för enheten.

netmask addr

specificerar nätmasken för nätverket som maskinen tillhör.

irq addr

denna parameter fungerar endast för viss hårdvara och sätter IRQ för enheten.

[- broadcast [addr]]

slår på mottagning av datagram som skickas till broadcastadressen, eller slår av mottagning av dessa datagram.

[- pointopoint [addr]]

sätter adressen på maskinen på andra änden av en punkt till punkt förbindelse som tex *slip* eller *ppp*.

hw <type> <addr>

sätter hårdvaruadressen för nätverksenheten. Oftast är detta inte användbart för Ethernet, men är användbart för andra typer av nätverk som tex AX.25.

Man kan använda kommandot *ifconfig* på alla nätverksgränssnitt. Vissa användarprogram, tex *pppd* och *dip*, konfigurerar enheterna automatisk när de skapar dem, så manuell användning av *ifconfig* är inte nödvändig.

5.5 Att konfigurera din Name Resolver.

*Name Resolver*n är en del av Linux standardbibliotek. Dess huvudsakliga funktion är att tillhandahålla en tjänst för att översätta människovänliga datornamn som `ftp.funet.fi` till maskinvänliga IP-adresser som `128.214.248.6`.

5.5.1 Vad består ett namn av?

De flesta är antagligen bekanta med hur datornamn uppträder i Internet, men man kanske inte vet hur de är konstruerade, eller rekonstruerade. Internet domännamn är av hierarkisk karaktär, dvs de har trädstruktur. En *domän* är en familj, eller grupp av namn. En *domän* kan brytas ned i *subdomäner*. En *toppdomän* är en domän som inte är en subdomän. Toppdomänerna är specificerade i RFC-920. Några exempel på de vanligaste topdomänerna är:

COM

Kommersiella Organisationer

EDU

Utbildningsorganisationer

GOV

Regeringsorganisationer

MIL

Militärorganisationer

ORG

Andra Organisationer

NET

Internet-Relaterade Organisationer

Landskod

Detta är koder som består av två bokstäver och som representerar ett visst land.

Alla dessa toppdomäner har subdomäner. Toppdomänerna baserade på landsnamn bryts sedan ned i subdomäner baserade på `com`, `edu`, `gov`, `mil` och `org` domäner (detta är ju dock inte fallet i .se-domänen ännu. Översättarens kommentar). Så till exempel blir det `com.au` och `gov.au` för kommersiella och regeringsorganisationer i Australien. Av historiska skäl så används de domäner som tillhör de icke landsbaserade toppdomänerna mestadels av organisationer i USA, trots att USA har sin egen landskod `.us`.

Nästa nedbrytningsnivå representerar oftast namnet på organisationen. Ytterligare subdomäner varierar i karaktär, oftast så baseras nästa nivå på avdelningar inom organisationen, men den kan baseras på vilket kriterium som helst som har någon betydelse för nätverksadministratörerna i organisationen.

Delen längst till vänster av namnet är alltid det unika namnet som värddatorn har och det kallas för *hostname*, den delen av namnet som finns till höger om *hostname* kallas för *domainname* och hela namnet heter *Fully Qualified Domain Name*.

Med min egen emailvärddator som exempel, så är 'Fully Qualified Domain Name' `perf.no.itg.telstra.com.au`. Detta betyder att hostname är `perf` och domainname är `no.itg.telstra.com.au`. Domännamnet är baserat på en toppdomän vilken är baserad på mitt land, Australien och eftersom min epostadress tillhör en komersiell organisation så har vi `.com` på nästa nivå. Företagets namn är (var) `telstra` och vår interna namnstruktur är baserad på den organisationella strukturen, i mitt fall så tillhör maskinen Information Technology Group, Network Operations sektionen.

5.5.2 Vilken information behöver man.

Man behöver veta vilken domän datorns namn skall tillhöra. Name Resolvermjukvaran tillhandahåller namnöversättning genom att göra förfrågningar till en *Domain Name Server*, så man behöver veta IP-adressen till en lokal namnserver som man kan använda.

Det finns tre filer som man behöver editera, jag berättar om dem i tur och ordning.

5.5.3 /etc/resolv.conf

Filen `/etc/resolv.conf` är huvudfilen för att konfigurera namnöversättaren. Dess format är ganska enkelt. Det är en textfil med ett nyckelord per rad. Det finns tre nyckelord som vanligtvis används, de är:

domain
: detta nyckelord specificerar det lokala domännamnet.

search
: detta nyckelord specificerar en lista med alternativa domännamn att leta i efter ett datornamn.

nameserver
: detta nyckelord, som kan användas flera gånger, specificerar en IP-adress till en namnserver (*Domain Name Server*) att använda när man översätter namn.

Ett exempel på `/etc/resolv.conf` kan se ut någonting som följande:

```
domain maths.wu.edu.au
search maths.wu.edu.au wu.edu.au
nameserver 192.168.10.1
nameserver 192.168.12.1
```

Detta exempel anger att man lägger till domännamnet `maths.wu.edu.au` till datornamn som ges utan domännamn, och om datorn inte hittas där så skall man leta även i `wu.edu.au` domänen. Två namnservrar är specificerade, som båda kan kontaktas av namnöversättaren när ett namn skall översättas.

5.5.4 /etc/host.conf

I filen `/etc/host.conf` kan man konfigurera hur namnöversättaren beter sig. Formatet för filen är beskrivet i detalj i manualsidan `resolv+`. I nästan alla fall fungerar följande exempel:

```
order hosts,bind
multi on
```

Denna konfiguration talar om för översättaren att titta i filen `/etc/hosts` innan den försöker göra en förfrågning till namnservern och att returnera alla giltiga adresser för en dator som den hittar i `/etc/hosts` och inte bara den första.

5.5.5 /etc/hosts

I filen `/etc/hosts` kan man lägga namn och IP-adresser för lokala maskiner. Om man placerar ett datornamn med respektive IP-adress i denna fil så behöver man inte göra en förfrågan till namnservern för att få reda på dess IP-adress. Nackdelen med detta är att man måste hålla filen uppdaterad själv ifall IP-adresserna ändras. I ett väl underhållet system så finns endast rader med 'loopback' och lokala datornamn i denna fil.

```
# /etc/hosts
127.0.0.1       localhost loopback
192.168.0.1     this.host.name
```

Man kan ange mer än ett datornamn per rad som på den första raden i exemplet, vilket är standardutformningen för loopbackenheten.

5.6 Att konfigurera loopbackenheten.

`loopback`-enheten är ett särskilt gränssnitt med vilket man kan göra anslutningar till sig själv. Det finns olika anledningar varför man skulle vilja göra detta, till exempel så kanske man vill testa någon nätverksmjukvara utan att störa någon annan på sitt nätverk. Av tradition så har IP-adressen `127.0.0.1` reserverats speciellt för loopback. Så, oavsett vilken maskin man går till, om man öppnar en telnet-anslutning till `127.0.0.1` så blir man alltid ansluten till den lokala datorn.

Att konfigurera loopbackenheten är enkelt och man bör göra följande:

```
# ifconfig lo 127.0.0.1
# route add -host 127.0.0.1 lo
```

Vi pratar mer om *route* kommandot i nästa sektion.

5.7 Routing.

Routing är ett stort ämne. Det skulle vara lätt att skriva stora mängder text om detta. De flesta kommer att ha hyfsat enkla krav på routingen, men vissa har det inte. Jag kommer bara att behandla grunderna om routing. Om man är intresserad av mer detaljerad information så föreslår jag att man letar i referenserna som ges i början av detta dokument.

Låt oss börja med en definition. Vad är IP routing? Här följer den definition som jag använder:

> IP routing är processen där en dator med flera nätverksanslutningar avgör vart den skall skicka IP datagram som den har tagit emot.

Det kan vara bra att illustrera detta med ett exempel. Antag en typisk kontorsrouter, den skulle kunna ha en PPP-länk till Internet, ett antal Ethernetsegment som går till arbetsstationerna och en annan PPP-länk till ett annat kontor. När routern tar emot ett datagram på någon av dess nätverksanslutningar, så är routing mekanismen den använder för att avgöra till vilken av nätverksanslutningarna som den skall skicka vidare datagrammet. Enkla datorer behöver också använda routing, alla Internetdatorer har åtminstone två nätverksenheter, en är loopbackenheten som beskrivs ovan och den andra är den som den använder för att 'prata' med resten av nätverket, kanske ett Ethernetkort, kanske ett PPP- eller SLIP-gränssnitt.

Hur fungerar routing? Varje dator har en speciell lista med routingregler som kallas för routingtabellen (routing table). Denna tabell innehåller rader som vanligtvis innehåller åtminstone tre fält, det första är en destinationsadress, det andra är namnet på gränssnittet som datagrammet skall routas till och det tredje är IP-adressen på en annan maskin som skall ta datagrammet på sitt nästa steg genom nätverket. I Linux kan man se denna tabell genom att använda följande kommando:

```
# cat /proc/net/route
```

eller genom att använda något av följande kommandon:

```
# /sbin/route -n
# /bin/netstat -r
```

Routingprocessen är hyfsat enkel: ett inkommande datagram tas emot, destinationsadressen undersöks och jämförs med varje rad i tabellen. Den rad som bäst matchar adressen väljs och datagrammet skickas vidare till det specificerade gränssnittet. Om gateway-fältet är ifyllt så skickas datagrammet vidare till den maskinen via det specificerade gränssnittet, annars antas det att destinationsadressen finns på nätverket som gränssnittet är kopplat till.

För att manipulera tabellen använder man ett speciellt kommando. Detta kommando tar kommandoradsargument och konverterar dem till systemanrop vilka ber kärnan att lägga till, ta bort eller ändra rader i routingtabellen. Kommandot är *route*.

Ett enkelt exempel. Antag att man har ett Ethernetnätverk. Man har fått veta att det är ett klass C nätverk med nätverksadressen `192.168.1.0`. Man har fått IP-adressen `192.168.1.10` som man kan använda och man har fått veta att `192.168.1.1` är en router som är ansluten till Internet.

Det första steget är att konfigurera gränssnittet som det beskrivits ovan. Man skulle använda ett kommando som:

```
# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
```

Man behöver nu lägga till en rad i routingtabellen för att tala om för kärnan att datagram till alla adresser som passar `192.168.1.*` skall sändas till en Ethernet-enhet. Detta med ett kommando som:

```
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

Notera `-net` parametern som används för att tala om för routingprogrammet att denna rad är en nätverksroute. Det andra alternativet är en `-host` route som är en route som är till en specifik IP-adress.

Med denna route kan man upprätta IP-anslutningar till alla datorer på sitt Ethernet-segment. Men vad händer med alla IP-datorer som inte finns på ens Ethernet-segment?

Det skulle vara ett väldigt svårt jobb att behöva lägga till router till varje tänkbart nätverk, så det finns ett specialtrix som används för att förenkla denna uppgift. Trixet kallas för `default` route. `Default` routen passar varje möjlig destination, men dåligt, så om det finns någon annan rad som passar den önskade adressen bättre så kommer den att användas istället för `default` routen. Tanken med `default` routen är helt enkelt att man skall kunna säga "och allt annat skall skickas dit". Så i exemplet som vi följt så bör man använda en rad som:

```
# route add default gw 192.168.1.1 eth0
```

Parametern `gw` talar om för route kommandot att nästa parameter är IP-adressen, eller namnet, på en gateway eller router dit alla datagram som passar denna rad skall skickas för vidare routing.

Så exemplets kompletta konfiguration skulle se ut så här:

```
# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
# route add default gw 192.168.1.1 eth0
```

Om man tittar noga i sina `rc`-filer för nätverket så kommer man att hitta åtminstone en som liknar detta. Detta är en väldigt vanlig konfiguration.

Låt oss nu titta på en något mer komplicerad routingkonfiguration. Antag att vi konfigurerar routern som vi tittade på tidigare, med en PPP-länk till Internet och några LAN-segment med arbetsstationer på kontoret. Antag att routern har tre Ethernet-segment och en PPP-länk. Vår routingkonfiguration skulle likna följande:

```
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
# route add -net 192.168.2.0 netmask 255.255.255.0 eth1
# route add -net 192.168.3.0 netmask 255.255.255.0 eth2
# route add default ppp0
```

Varje arbetsstation skulle använda den enklare formen som beskrevs ovan, endast routern behöver ange alla nätverksrouter separat eftersom hos arbetsstationerna så tar `default` route mekanismen hand om allihop och låter routern sköta den 'riktiga' routingen. Man kanske undrar varför defaultrouten inte anger en `gw`. Anledningen är enkel, seriella länkprotokoll såsom PPP och SLIP har alltid endast två maskiner på sitt nätverk, en i varje ända. Att ange maskinen på andra sidan länken som gateway är meningslöst och redundant eftersom det inte finns något annat val, så man behöver inte ange någon gateway för den typen av nätverksanslutningar. Andra nätverkstyper såsom Ethernet, arcnet eller Token Ring kräver att en gateway specificeras eftersom dessa nätverken stödjer att ett stort antal datorer ansluts till dem.

5.7.1 Så vad gör programmet *routed*?

Routingkonfigurationen som beskrivs ovan passar bäst för enklare nätverksarrangemang där det endast finns en möjlig väg till varje destination. När man har ett mer komplext arrangemang så blir det lite besvärligare. Som tur är behöver de flesta inte bry sig om detta.

Det stora problemet med 'manuell routing' eller 'statisk routing' är att om en maskin eller länk fallerar i nätverket så kan man endast dirigera om sina datagram, om det finns någon annan väg, genom att manuellt ge lämpliga kommandon. Naturligtvis är detta klumpigt, långsamt, opraktiskt och felbenäget. Olika tekniker har utvecklats för att automatiskt ändra routingtabeller om det uppstår nätverksfel där det finns alternativa vägar att leda trafiken, alla dessa tekniker är löst samlade under termen 'dynamiska routing algoritmer'.

Man kanske har hört talas om några av de vanligare dynamiska routingalgorimerna. De allra vanligaste är RIP (Routing Information Protocol) och OSPF (Open Shortest Path First). RIP är väldigt vanlig i små nätverk som till exempel små/mellanstora företagsnätverk. OSPF är modernare och mer kapabel att hantera stora nätverk och bättre anpassad till omgivningar där det finns ett stort antal möjliga vägar genom ett nätverk. Vanliga implementeringar av dessa är: *routed* (RIP) och *gated* (RIP, OSPF och andra). Programmet *routed* finns normalt med i Linuxdistributioner eller finns inkluderat i 'NetKit' paketet ovan.

Ett exempel på var och hur man skulle kunna använda en dynamisk routingalgoritm skulle kunna se ut som följer:

```
192.168.1.0 /                192.168.2.0 /
  255.255.255.0                255.255.255.0
 -                            -
```

```
     |                                     |
     |   /-----\                 /-----\   |
     |   |     |ppp0   //   ppp0|     |   |
eth0 |---|  A  |------//---------|  B  |---| eth0
     |   |     |     //          |     |   |
     |   \-----/                 \-----/   |
     |      \ ppp1           ppp1 /        |
     -       \                   /         -
              \                 /
               \               /
                \             /
                 \           /
                  \         /
                   \       /
                    \     /
                     \   /
                  ppp0\   /ppp1
                     /-----\
                     |     |
                     |  C  |
                     |     |
                     \-----/
                        |eth0
                        |
                   |---------|
                   192.168.3.0 /
                      255.255.255.0
```

Vi har tre routrar A, B och C. Var och en har ett Ethernet-segment med ett klass C IP-nätverk (nätmask 255.255.255.0). Varje router har även en PPP-länk till var och en av de andra routrarna. Nätverket bildar en triangel.

Det bör vara klart att routingtabellen för router A skulle kunna se ut som:

```
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
# route add -net 192.168.2.0 netmask 255.255.255.0 ppp0
# route add -net 192.168.3.0 netmask 255.255.255.0 ppp1
```

Detta skulle fungera utmärkt ända tills länken mellan router A och B fallerar. Om så skulle ske så skulle datorer på A's Ethernet-segment inte kunna nå datorer på B's Ethernet-segment eftersom dess datagram skulle routas till A's ppp0-länk, vilken är trasig. De skulle fortfarande kunna 'prata' med datorer på C's Ethernet-segment och datorer på C's Ethernet-segment skulle kunna kontakta datorer på B's Ethernet-segment eftersom länkarna mellan A och C respektive mellan C och B fortfarande fungerar.

Men vänta lite nu, om A kan prata med C och C fortfarande kan prata med B, varför skulle inte A kunna routa sina datagram till B via C och låta C skicka dem till B? Detta är precis den typen av problem som dynamiska routingalgoritmer som RIP designades för att lösa. Om var och en av routrarna A, B och C körde en routingdaemon så skulle deras routingtabeller automatiskt uppdateras till att spegla det nya tillståndet i nätverket om någon av länkarna skulle fallera. För att konfigurera ett sådant nätverk är enkelt, på varje router behöver man endast göra två saker. I detta fall för router A:

```
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
# /usr/sbin/routed
```

Routingdaemonen *routed* hittar automatiskt alla aktiva nätverksportar när den startar och skickar och lyssnar efter meddelanden på var och en av nätverksenheterna så att den kan fastställa och uppdatera routingtabellen på datorn.

Detta har varit en väldigt översiktlig förklaring av dynamisk routing och var man bör använda det. Om man vill ha mer information så sök ibland referenserna i början av dokumentet.

De viktiga punkterna relaterade till dynamisk routing är:

1. Man behöver bara använda dynamisk routing när ens Linuxburk har möjligheten att välja flera olika routes till en destination.
2. Den dynamiska routingdaemonen kommer automatiskt att modifiera routingtabellen för att ställa in sig till förändringar i nätverket.
3. RIP är bra anpassat för små till medelstora nätverk.

5.8 Att konfigurera nätverksservrar och tjänster.

Nätverksservrar och tjänster är de program som låter en avlägsen användare använda sig av ens Linuxburk. Serverprogram lyssnar på nätverksportar. Nätverksportar är hjälpmedel för att adressera en speciell tjänst på en dator och är hur en server vet skillnaden mellan en inkommande telnet-anslutning och en inkommande ftp-anslutning. Den avlägsna användaren upprättar en nätverksanslutning till Linuxburken och serverprogrammet, nätverksdaemonen, lyssnar på den porten och accepterar anslutningen och exekverar. Det finns två sätt på vilka nätverksdaemoner kan fungera. Båda är flitigt använda i praktiken. De två sätten är:

fristående

nätverksdaemonen lyssnar på den angivna porten och när den upptäcker en inkommande anslutning så sköter den om anslutningen själv för att vidare tillhandahålla tjänster.

slav till *inetd* servern

inetd servern är en särskild nätverksdaemon som är specialiserad på att ta hand om inkommande nätvarksanslutningar. Den har en konfigurationsfil som talar om vilket program som skall köras då en inkommande anslutning tas emot. Varje port kan konfigureras för något av protokollen TCP eller UDP. Portarna är beskrivna i en annan fil som vi skall prata om snart.

Det finns två viktiga filer som behöver konfigureras. De är `/etc/services` som tilldelar namn till portnummer och `/etc/inetd.conf` som är konfigurationsfilen till nätverksdaemonen *inetd*.

5.8.1 /etc/services

Filen `/etc/services` är en enkel databas som associerar ett människovänligt namn med en maskinvänlig port. Formatet är ganska enkelt. Filen är en textfil där varje rad representerar en rad i databasen. Varje rad består av tre fält som är separerade av valfritt antal vita tecken (tab eller space). Fälten är:

```
name      port/protocol       aliases     # comment
```

name

ett namn, bestående av ett ord, som representerar tjänsten som skall beskrivas.

port/protocol

detta fält är indelat i två delfält.

port

ett nummer som anger portnummret som den namngivna tjänsten kommer att vara tillgänglig på. De flesta vanliga tjänster har tilldelats reserverade nummer. Dessa finns beskrivna i `RFC-1340`.

protocol

detta delfält kan sättas till antingen `tcp` eller `udp`.

Det är viktigt att notera att en rad med `18/tcp` är väldigt olik en rad med `18/udp` och att det inte finns någon teknisk anledning till att samma tjänst finns tillgänglig på båda. Normalt råder sunt förnuft och det är bara om en viss tjänst finns tillgänglig via både `tcp` och `udp` som man kommer att se rader med båda.

aliases

andra namn som kan användas för att referera till denna tjänsten.

All text som finns efter ett `#` tecken på en rad ignoreras och behandlas som en kommentar.

Ett exempel på en `/etc/services` fil. Alla moderna Linuxdistributioner kommer med en bra `/etc/services` fil. Men i fall man råkar bygga ett system från grunden så är här en kopia av `/etc/services` filen som kommer med *Debian* `<http://www.debian.org/>` distributionen.

```
# /etc/services:
# $Id: services,v 1.3 1996/05/06 21:42:37 tobias Exp $
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1340, ``Assigned Numbers'' (July 1992).  Not all ports
# are included, only the more common ones.

tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message send protocol
msp             18/udp                          # message send protocol
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
ssh             22/tcp                          # SSH Remote Login Protocol
ssh             22/udp                          # SSH Remote Login Protocol
telnet          23/tcp
# 24 - private
smtp            25/tcp          mail
# 26 - unassigned
```

```
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/udp          resource        # resource location
nameserver      42/tcp          name            # IEN 116
whois           43/tcp          nicname
re-mail-ck      50/tcp                          # Remote Mail Checking Protocol
re-mail-ck      50/udp                          # Remote Mail Checking Protocol
domain          53/tcp          nameserver      # name-domain server
domain          53/udp          nameserver
mtp             57/tcp                          # deprecated
bootps          67/tcp                          # BOOTP server
bootps          67/udp
bootpc          68/tcp                          # BOOTP client
bootpc          68/udp
tftp            69/udp
gopher          70/tcp                          # Internet Gopher
gopher          70/udp
rje             77/tcp          netrjs
finger          79/tcp
www             80/tcp          http            # WorldWideWeb HTTP
www             80/udp                          # HyperText Transfer Protocol
link            87/tcp          ttylink
kerberos        88/tcp          kerberos5 krb5  # Kerberos v5
kerberos        88/udp          kerberos5 krb5  # Kerberos v5
supdup          95/tcp
# 100 - reserved
hostnames       101/tcp         hostname        # usually from sri-nic
iso-tsap        102/tcp         tsap            # part of ISODE.
csnet-ns        105/tcp         cso-ns          # also used by CSO name server
csnet-ns        105/udp         cso-ns
rtelnet         107/tcp                         # Remote Telnet
rtelnet         107/udp
pop-2           109/tcp         postoffice      # POP version 2
pop-2           109/udp
pop-3           110/tcp                         # POP version 3
pop-3           110/udp
sunrpc          111/tcp         portmapper      # RPC 4.0 portmapper TCP
sunrpc          111/udp         portmapper      # RPC 4.0 portmapper UDP
auth            113/tcp         authentication tap ident
sftp            115/tcp
uucp-path       117/tcp
nntp            119/tcp         readnews untp   # USENET News Transfer Protocol
ntp             123/tcp
ntp             123/udp                         # Network Time Protocol
netbios-ns      137/tcp                         # NETBIOS Name Service
netbios-ns      137/udp
netbios-dgm     138/tcp                         # NETBIOS Datagram Service
netbios-dgm     138/udp
netbios-ssn     139/tcp                         # NETBIOS session service
netbios-ssn     139/udp
imap2           143/tcp                         # Interim Mail Access Proto v2
imap2           143/udp
snmp            161/udp                         # Simple Net Mgmt Proto
snmp-trap       162/udp         snmptrap        # Traps for SNMP
cmip-man        163/tcp                         # ISO mgmt over IP (CMOT)
```

```
cmip-man        163/udp
cmip-agent      164/tcp
cmip-agent      164/udp
xdmcp           177/tcp                         # X Display Mgr. Control Proto
xdmcp           177/udp
nextstep        178/tcp         NeXTStep NextStep       # NeXTStep window
nextstep        178/udp         NeXTStep NextStep       # server
bgp             179/tcp                         # Border Gateway Proto.
bgp             179/udp
prospero        191/tcp                         # Cliff Neuman's Prospero
prospero        191/udp
irc             194/tcp                         # Internet Relay Chat
irc             194/udp
smux            199/tcp                         # SNMP Unix Multiplexer
smux            199/udp
at-rtmp         201/tcp                         # AppleTalk routing
at-rtmp         201/udp
at-nbp          202/tcp                         # AppleTalk name binding
at-nbp          202/udp
at-echo         204/tcp                         # AppleTalk echo
at-echo         204/udp
at-zis          206/tcp                         # AppleTalk zone information
at-zis          206/udp
z3950           210/tcp         wais            # NISO Z39.50 database
z3950           210/udp         wais
ipx             213/tcp                         # IPX
ipx             213/udp
imap3           220/tcp                         # Interactive Mail Access
imap3           220/udp                         # Protocol v3
ulistserv       372/tcp                         # UNIX Listserv
ulistserv       372/udp
#
# UNIX specific services
#
exec            512/tcp
biff            512/udp         comsat
login           513/tcp
who             513/udp         whod
shell           514/tcp         cmd             # no passwords used
syslog          514/udp
printer         515/tcp         spooler         # line printer spooler
talk            517/udp
ntalk           518/udp
route           520/udp         router routed   # RIP
timed           525/udp         timeserver
tempo           526/tcp         newdate
courier         530/tcp         rpc
conference      531/tcp         chat
netnews         532/tcp         readnews
netwall         533/udp                         # -for emergency broadcasts
uucp            540/tcp         uucpd           # uucp daemon
remotefs        556/tcp         rfs_server rfs  # Brunhoff remote filesystem
klogin          543/tcp                         # Kerberized `rlogin' (v5)
kshell          544/tcp         krcmd           # Kerberized `rsh' (v5)
kerberos-adm    749/tcp                         # Kerberos `kadmin' (v5)
```

```
#
webster         765/tcp                         # Network dictionary
webster         765/udp
#
# From ``Assigned Numbers'':
#
#> The Registered Ports are not controlled by the IANA and on most systems
#> can be used by ordinary user processes or programs executed by ordinary
#> users.
#
#> Ports are used in the TCP [45,106] to name the ends of logical
#> connections which carry long term conversations.  For the purpose of
#> providing services to unknown callers, a service contact port is
#> defined.  This list specifies the port used by the server process as its
#> contact port.  While the IANA can not control uses of these ports it
#> does register or list uses of these ports as a convienence to the
#> community.
#
ingreslock      1524/tcp
ingreslock      1524/udp
prospero-np     1525/tcp                # Prospero non-privileged
prospero-np     1525/udp
rfe             5002/tcp                # Radio Free Ethernet
rfe             5002/udp                # Actually uses UDP only
bbs             7000/tcp                # BBS service
#
#
# Kerberos (Project Athena/MIT) services
# Note that these are for Kerberos v4 and are unofficial.  Sites running
# v4 should uncomment these and comment out the v5 entries above.
#
kerberos4       750/udp         kdc     # Kerberos (server) udp
kerberos4       750/tcp         kdc     # Kerberos (server) tcp
kerberos_master 751/udp                 # Kerberos authentication
kerberos_master 751/tcp                 # Kerberos authentication
passwd_server   752/udp                 # Kerberos passwd server
krb_prop        754/tcp                 # Kerberos slave propagation
krbupdate       760/tcp         kreg    # Kerberos registration
kpasswd         761/tcp         kpwd    # Kerberos "passwd"
kpop            1109/tcp                # Pop with Kerberos
knetd           2053/tcp                # Kerberos de-multiplexor
zephyr-srv      2102/udp                # Zephyr server
zephyr-clt      2103/udp                # Zephyr serv-hm connection
zephyr-hm       2104/udp                # Zephyr hostmanager
eklogin         2105/tcp                # Kerberos encrypted rlogin
#
# Unofficial but necessary (for NetBSD) services
#
supfilesrv      871/tcp                 # SUP server
supfiledbg      1127/tcp                # SUP debugging
#
# Datagram Delivery Protocol services
#
rtmp            1/ddp                   # Routing Table Maintenance Protocol
nbp             2/ddp                   # Name Binding Protocol
```

```
echo            4/ddp                   # AppleTalk Echo Protocol
zip             6/ddp                   # Zone Information Protocol
#
# Debian GNU/Linux services
rmtcfg          1236/tcp                # Gracilis Packeten remote config server
xtel            1313/tcp                # french minitel
cfinger         2003/tcp                # GNU Finger
postgres        4321/tcp                # POSTGRES
mandelspawn     9359/udp        mandelbrot      # network mandelbrot

# Local services
```

5.8.2 /etc/inetd.conf

Filen `/etc/inetd.conf` är konfigurationsfilen för serverdaemonen *inetd*. Dess funktion är att tala om för *inetd* vad som skall hända när den tar emot en anslutningsförfrågan för en viss tjänst. För varje tjänst för vilken man önskar acceptera anslutningar måste man tala om för *inetd* vilken nätverksserver den skall starta och hur den skall startas.

Dess format är ganska enkelt. Det är en textfil där varje rad beskriver en tjänst som man vill tillhandahålla. All text efter ett `#` tecken på en rad ignoreras och ses som en kommentar. Varje rad innehåller sju fält som är separerade av valfritt antal vita tecken (tab eller space). Det generella formatet är som följer:

```
service  socket_type  proto  flags  user  server_path  server_args
```

service

är tjänsten som är relevant för denna konfiguration som den benämns i `/etc/services`.

socket_type

detta fält beskriver vilken typ av socket som denna rad kommer att beteckna som relevant, tillåtna värden är `stream`, `dgram`, `raw`, `rdm` eller `seqpacket`. Detta är lite tekniskt till karaktären men som tumregel så använder nästan alla `tcp`-baserade tjänster `stream` och alla `udp`-baserade tjänster `dgram`. Det är bara väldigt speciella typer av daemoner som använder de andra värdena.

proto

protokollet som skall anses som giltigt för denna rad. Detta skall passa ihop med lämplig rad i `/etc/services` och är vanligtvis antingen `tcp` eller `udp`. Sun RPC (Remote Procedure Call) baserade tjänster använder `rpc/tcp` eller `rcp/udp`.

flags

det finns egentligen bara två möjliga värden för detta fält. Detta fält talar om för `inetd` om serverprogramvaran stänger socketen efter det att den startats och därför om *inetd* kan starta en annan server på nästa anslutningsförfrågan. Återigen så är detta lite besvärligt att lista ut, men som tumregel så skall alla `tcp` servrar ha värdet `nowait` och de flesta `udp` servrar `wait`. Observera att det finns undantag för detta, så man bör endast följa exemplet om man inte vet.

user

detta fält talar om vilket användarkonto från `/etc/passwd` som skall sättas som ägare till nätverksdaemonen när den startas. Detta är ofta användbart för att skydda sig mot säkerhetsrisker. Man kan sätta värdet till `nobody` så att skadan minimeras ifall nätverksserverns säkerhet fallerar. Men vanligtvis så är detta fält satt till `root` eftersom många servrar måste ha root-rättigheter för att fungera korrekt.

server_path

detta fält skall innehålla det absoluta filnamnet till serverprogrammet som skall exekveras för denna raden.

server_args

detta fältet utgör resten av raden och är valfri. Det är här som man placerar kommandoradsargument som man vill skicka med till serverdaemonen när den startas.

Ett exempel på en `/etc/inetd.conf` fil. Som för filen `/etc/services` så inkluderar alla moderna distributioner en bra `/etc/inetd.conf` fil som man kan arbeta med. För att vara komplett så inkluderas `/etc/inetd.conf` från *Debian* <http://www.debian.org/> distributionen.

```
# /etc/inetd.conf:  see inetd(8) for further informations.
#
# Internet server configuration database
#
#
# Modified for Debian by Peter Tobias <tobias@et-inf.fho-emden.de>
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
# Internal services
#
#echo           stream  tcp     nowait  root    internal
#echo           dgram   udp     wait    root    internal
discard         stream  tcp     nowait  root    internal
discard         dgram   udp     wait    root    internal
daytime         stream  tcp     nowait  root    internal
daytime         dgram   udp     wait    root    internal
#chargen        stream  tcp     nowait  root    internal
#chargen        dgram   udp     wait    root    internal
time            stream  tcp     nowait  root    internal
time            dgram   udp     wait    root    internal
#
# These are standard services.
#
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.telnetd
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.ftpd
#fsp    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.fspd
#
# Shell, login, exec and talk are BSD protocols.
#
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd
login   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind
#exec   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rexecd
talk    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.talkd
ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.ntalkd
#
# Mail, news and uucp services.
#
smtp    stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.smtpd
#nntp   stream  tcp     nowait  news    /usr/sbin/tcpd  /usr/sbin/in.nntpd
#uucp   stream  tcp     nowait  uucp    /usr/sbin/tcpd  /usr/lib/uucp/uucico
#comsat dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.comsat
```

```
#
# Pop et al
#
#pop-2  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop2d
#pop-3  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop3d
#
# 'cfinger' is for the GNU finger server available for Debian.  (NOTE: The
# current implementation of the 'finger' daemon allows it to be run as 'root'.)
#
#cfinger stream tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.cfingerd
#finger stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.fingerd
#netstat        stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/netstat
#systat stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/ps -auwwx
#
# Tftp service is provided primarily for booting.  Most sites
# run this only on machines acting as "boot servers."
#
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd /boot
#bootps dgram   udp     wait    root    /usr/sbin/bootpd        bootpd -i -t 120
#
# Kerberos authenticated services (these probably need to be corrected)
#
#klogin         stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind -k
#eklogin        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind -k -x
#kshell         stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd -k
#
# Services run ONLY on the Kerberos server (these probably need to be corrected)
#
#krbupdate      stream tcp      nowait  root    /usr/sbin/tcpd  /usr/sbin/registerd
#kpasswd        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/kpasswdd
#
# RPC based services
#
#mountd/1       dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.mountd
#rstatd/1-3     dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rstatd
#rusersd/2-3    dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rusersd
#walld/1        dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rwalld
#
# End of inetd.conf.
ident           stream  tcp     nowait  nobody  /usr/sbin/identd        identd -i
```

5.9 Andra nätverksrelaterade konfigurationsfiler.

Det finns ett antal andra filer som är relaterade till nätverkskonfiguration i Linux som man skulle kunna vara intresserad av. Man behöver kanske aldrig modifiera dessa filer, men det är värt att beskriva dem ändå så att man vet vad de innehåller och vad de används till.

5.9.1 `/etc/protocols`

Filen `/etc/protocols` är en databas som mappar protokollens id-nummer mot protokollens namn. Detta används av programmerare för att låta dem ange protokoll med dess namn i program, och även av vissa program, som till exempel *tcpdump*, för att kunna skriva ut namn istället för nummer. Syntaxen för filen är:

```
protocolname  number  aliases
```

I *Debian* <http://www.debian.org/> distributionen ser `/etc/protocols` ut som följande:

```
# /etc/protocols:
# $Id: protocols,v 1.1 1995/02/24 01:09:41 imurdock Exp $
#
# Internet (IP) protocols
#
#       from: @(#)protocols     5.1 (Berkeley) 4/17/89
#
# Updated for NetBSD based on RFC 1340, Assigned Numbers (July 1992).

ip      0       IP              # internet protocol, pseudo protocol number
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # Internet Group Management
ggp     3       GGP             # gateway-gateway protocol
ipencap 4       IP-ENCAP        # IP encapsulated in IP (officially ``IP'')
st      5       ST              # ST datagram mode
tcp     6       TCP             # transmission control protocol
egp     8       EGP             # exterior gateway protocol
pup     12      PUP             # PARC universal packet protocol
udp     17      UDP             # user datagram protocol
hmp     20      HMP             # host monitoring protocol
xns-idp 22      XNS-IDP         # Xerox NS IDP
rdp     27      RDP             # "reliable datagram" protocol
iso-tp4 29      ISO-TP4         # ISO Transport Protocol class 4
xtp     36      XTP             # Xpress Tranfer Protocol
ddp     37      DDP             # Datagram Delivery Protocol
idpr-cmtp       39      IDPR-CMTP       # IDPR Control Message Transport
rspf    73      RSPF            # Radio Shortest Path First.
vmtp    81      VMTP            # Versatile Message Transport
ospf    89      OSPFIGP         # Open Shortest Path First IGP
ipip    94      IPIP            # Yet Another IP encapsulation
encap   98      ENCAP           # Yet Another IP encapsulation
```

5.9.2 /etc/networks

Filen `/etc/networks` har en liknande funktion som filen `/etc/hosts`. Den tillhandahåller en enkel databas av nätverksnamn som mappas mot nätverksadresser. Dess format skiljer sig genom att det får endast finnas två fält per rad och att fälten skrivs som:

```
networkname networkaddress
```

Ett exempel kam se ut såhär:

```
loopnet    127.0.0.0
localnet   192.168.0.0
amprnet    44.0.0.0
```

När man använder kommandon som *route*, om en destination är ett nätverk och det nätverket finns i `/etc/networks`, så kommer *route* att visa nätverksnamnet istället för dess adress.

5.10 Nätverkssäkerhet och åtkomstkontroll.

Låt mig börja denna sektion med att varna för att säkra en maskin och ett nätverk mot illvilliga attacker är en komplex konst. Jag anser mig inte själv vara en expert på detta område och även om de följande mekanismerna som jag beskriver kommer att hjälpa, så om man är riktigt allvarlig när det gäller säkerhet så rekommenderar jag att man gör egna undersökningar i ämnet. Det finns många bra referenser på Internet relaterade till detta ämne.

En viktig tumregel är: '**Kör inte servrar som inte skall användas**'. Många distributioner kommer konfigurerade med alla möjliga tjänster som startas automatiskt. För att säkerställa en miniminivå av säkerhet så bör man gå igenom sin `/etc/inetd.conf` och kommentera bort (*sätt ett '#' i början av raden*) alla rader som innehåller tjänster vilka man inte tänker använda. Bra kandidater är tjänster såsom: `shell`, `login`, `exec`, `uucp`, `ftp` och informativa tjänster som `finger`, `netstat` och `systat`.

Det finns alla möjliga sorters säkerhets- och åtkomstkontrollmekanismer, jag kommer att beskriva de mest elementära av dem.

5.10.1 /etc/ftpusers

Filen `/etc/ftpusers` är en enkel mekanism med vilken man kan vägra vissa användare att logga in till maskinen via ftp. Filen `/etc/ftpusers` läses av ftp-daemonen (*ftpd*) när en inkommande ftp-anslutning tas emot. Filen är en enkel lista av de användare som inte är tillåtna att logga in. Den skulle kunna se ut som:

```
# /etc/ftpusers - users not allowed to login via ftp
root
uucp
bin
mail
```

5.10.2 /etc/securetty

I filen `/etc/securetty` kan man specificera vilka `tty` enheter som `root` är tillåten att logga in på. Filen `/etc/securetty` läses av loginprogrammet (vanligtvis */bin/login*). Dess format är en lista av de tty enhetsnamn som är tillåtna, på alla andra är `root` login otillåten:

```
# /etc/securetty - tty's on which root is allowed to login
tty1
tty2
tty3
tty4
```

5.10.3 *tcpd* åtkomstkontrollmekanism.

Programmet *tcpd* som man sett listat i `/etc/inetd.conf` tillhandahåller loggning och åtkomstkontrollmekanismer för tjänster som det är konfigurerat att skydda.

När det aktiveras av programmet *inetd* så läser det två filer som innehåller åtkomstregler och antingen tillåter det eller nekar åtkomst till servern som det skyddar.

Programmet söker i reglerna tills dess att det hittar det första mönstret som passar. Hittar det inget passande mönster så antas det att åtkomst skall tillåtas till vem som helst. Filerna som söks igenom i sekvens är: `/etc/hosts.allow`, `/etc/hosts.deny`. Jag kommer att beskriva dem i ordning. För en komplett beskrivning av detta så bör man titta i lämpliga *manualblad* (`hosts_access(5)` är ett bra ställe att börja på).

/etc/hosts.allow Filen `/etc/hosts.allow` är en konfigurationsfil för programmet */usr/sbin/tcpd*. Filen innehåller regler som beskriver vilka datorer som är *tillåtna* åtkomst till en tjänst på maskinen.

Filformatet är ganska enkelt:

```
# /etc/hosts.allow
#
# <service list>: <host list> [: command]
```

`service list`

är en kommaseparerad lista av servernamn som denna regeln gäller för. Exempel: `ftpd`, `telnetd` och `fingerd`.

`host list`

är en kommaseparerad lista av datornamn. Man kan även använda IP-adresser. Man kan dessutom ange datornamn eller adresser med hjälp av 'wildcards' för att täcka grupper av datorer. Exempel: `gw.vk2ktj.ampr.org` för att täcka en enskild dator, `.uts.edu.au` för att täcka alla datornamn som slutar med den strängen, `44.` för att täcka alla IP-adresser som börjar med de siffrorna. Det finns några särskilda tokens för att förenkla konfigurationen, några av dessa är: `ALL` täcker alla datorer, `LOCAL` täcker alla datornamn som inte innehåller en '`.`' dvs som är i samma domän som din maskin och `PARANOID` täcker alla datorer vars namn inte stämmer överens med sin adress (name spoofing). Det finns en sista användbar token. `EXCEPT` tillåter dig att ange en lista med undantag. Detta visas i ett exempel senare.

`command`

är en valfri parameter. Detta är det absoluta filnamnet för ett kommando som skall exekveras varje gång denna regel används. Det skulle till exempel kunna köra ett program som försöker identifiera vem som är påloggad på den anslutande datorn, eller att generera ett mail eller någon annan varning till en systemadministratör att någon försöker ansluta. Det finns ett antal utökningar som kan inkluderas, till exempel: `%h` expanderar till namnet på den anslutande datorn eller adress om den inte har något namn, `%d` daemonnamnet anropas.

Ett exempel:

```
# /etc/hosts.allow
#
# Allow mail to anyone
in.smtpd: ALL
# All telnet and ftp to only hosts within my domain and my host at home.
telnetd, ftpd: LOCAL, myhost.athome.org.au
# Allow finger to anyone but keep a record of who they are.
fingerd: ALL: (finger @%h | mail -s "finger from %h" root)
```

/etc/hosts.deny Filen `/etc/hosts.deny` är en konfigurationsfil för programmet */usr/sbin/tcpd*. Filen innehåller regler som beskriver vilka datorer som är *nekade* åtkomst till en tjänst på maskinen.

Ett enkelt exempel:

```
# /etc/hosts.deny
#
# Disallow all hosts with suspect hostnames
ALL: PARANOID
#
```

```
# Disallow all hosts.
ALL: ALL
```

Raden med `PARANOID` är egentligen redundant eftersom den andra raden fäller allt i vilket fall. Någon av dessa rader skulle vara tänkbara beroende på vilka krav man har.

Att ha `ALL: ALL` i `/etc/hosts.deny` och sedan specifikt ange de tjänster och datorer som man vill ha i filen `/etc/hosts.allow` är den säkraste konfigurationen.

5.10.4 /etc/hosts.equiv

Filen `hosts.equiv` används för att ge vissa datorer och användare åtkomsträttigheter till konton på maskinen utan att behöva ange ett lösenord. Detta är användbart i en säker omgivning där man kontrollerar alla maskinerna, men det är en säkerhetsrisk i annat fall. Datorn är bara så säker som den minst säkra av de datorer man litar på. För att maximera säkerheten så bör man inte använda denna mekanismen och påverka sina användare att inte använda filen `.rhosts` heller.

5.10.5 Konfigurera *ftp*-daemonen ordentligt.

Många sajter är intresserade av att köra en anonym *ftp* server för att tillåta andra personer att ladda upp och ladda ner filer utan att ha ett särskilt användarid. Om man bestämmer sig för att tillhandahålla denna tjänst så skall man se till att man konfigurerar sin *ftp*-daemon ordentligt för anonym åtkomst. De flesta *manualblad* för `ftpd(8)` beskriver hur man skall göra detta. Man bör alltid försäkra sig om att man följer dessa instruktioner. Ett viktigt tips är att inte använda en kopia av sin `/etc/passwd` fil i `/etc` katalogen för det anonyma kontot, se till att man tar bort alla detaljer om konton som man inte måste ha, annars kommer man att vara sårbar mot tekniker för lösenordscracking.

5.10.6 Brandväggar.

Att inte tillåta datagram att ens nå fram till din maskin eller servrar är ett utmärkt sätt att säkra systemet. Detta beskrivs i *Firewall-HOWTO* `<Firewall-HOWTO.html>`.

5.10.7 Andra förslag.

Här är några andra, potentiellt religiösa förslag som man kan tänka på.

sendmail

oavsett dess popularitet så framträder *sendmail* daemonen med skrämmande jämna mellanrum på säkerhetsmeddelanden. Det är upp till en själv om man väljer att köra den.

NFS och andra Sun RPC tjänster

man bör vara aktsam med dessa. Det finns alla möjliga olika sätt att utnyttja dessa tjänster. Det är svårt att hitta alternativ till en tjänst som NFS, men om man konfigurerar dem så skall man se till att vara försiktig med vem man ger mount-rättigheter till.

6 Nätverksspecifik Information.

Följande delsektioner är specifika för vissa nätverkstekniker. Informationen i dessa sektioner gäller inte nödvändigtvis för någon annan nätverksteknik.

6.1 ARCNet

Enhetsnamn för ARCNet är ‘arc0e’, ‘arc1e’, ‘arc2e’ osv eller ‘arc0s’, ‘arc1s’, ‘arc2s’ osv. Det första kortet som hittas av kärnan tilldelas ‘arc0e’ eller ‘arc0s’ och resten tilldelas namn sekvensiellt i den ordning som de hittas. Bokstaven på slutet av namnet betecknar om man har valt Ethenet-inkapsling som paketformat eller RFC1051 som paketformat.

Kompileringsalternativ för Kärnan:

```
Network device support  --->
    [*] Network device support
    <*> ARCnet support
    [ ]   Enable arc0e (ARCnet "Ether-Encap" packet format)
    [ ]   Enable arc0s (ARCnet RFC1051 packet format)
```

När man väl har en kärna som stöder sitt Ethernetkort så är konfiguration av kortet lätt.

Man skulle kunna använda något som liknar:

```
# ifconfig arc0e 192.168.0.1 netmask 255.255.255.0 up
# route add -net 192.168.0.0 netmask 255.255.255.0 arc0e
```

Titta gärna i filerna /usr/src/linux/Documentation/networking/arcnet.txt och /usr/src/linux/Documentation/networking/arcnet-hardware.txt för mer information.

ARCNet-stöd utvecklades av Avery Pennarun, apenwarr@foxnet.net.

6.2 Appletalk (AF_APPLETALK)

Appletalk har inga särskilda enhetsnamn eftersom det använder existerande nätverksenheter.

Kompileringsalternativ för Kärnan:

```
Networking options  --->
    <*> Appletalk DDP
```

Appletalk-stöd gör det möjligt för Linuxburken att kommunicera med Apple-nätverk. Ett viktigt användningsområde för detta är möjligheten att kunna dela resurser, som till exempel skrivare och hårddiskar, mellan Linux- och Appledatorer. Man behöver ytterligare programvara, *netatalk*, för detta. Wesley Craig, netatalk@umich.edu, representerar en grupp som heter ‘Research Systems Unix Group’ på University of Michigan och de har utvecklat *netatalk*-paketet som tillhandahåller programvara som implementerar protokollstacken för Appletalk och några andra användbara verktyg. Paketet *netatalk* finns antingen med i Linuxditributionen, eller så kan man ladda hem det via ftp från *University of Michigan* <ftp://terminator.rs.itd.umich.edu/unix/netatalk/>

För att kompilera och installera paketet gör man ungefär så här:

```
# cd /usr/src
# tar xvfz .../netatalk-1.4b2.tar.Z
- You may want to edit the ‘Makefile’ at this point, specifically to change
  the DESTDIR variable which defines where the files will be installed later.
  The default of /usr/local/atalk is fairly safe.
# make
- as root:
# make install
```

6.2.1 Att konfigurera mjukvaran för Appletalk.

Det första man måste göra för att få det att fungera är att se till att de rätta raderna finns med i filen `/etc/services`. Raderna man behöver är:

```
rtmp    1/ddp   # Routing Table Maintenance Protocol
nbp     2/ddp   # Name Binding Protocol
echo    4/ddp   # AppleTalk Echo Protocol
zip     6/ddp   # Zone Information Protocol
```

Nästa steg är att skapa konfigurationsfiler för Appletalk i katalogen `/usr/local/atalk/etc` (eller var man nu installerade paketet).

Den första filen som behövs är `/usr/local/atalk/etc/atalkd.conf`. Till en början behöver denna fil endast en rad som anger namnet på den nätverksenhet som är kopplad nätverket där Applemaskinerna finns:

```
eth0
```

Daemonen för Appletalk kommer att lägga dit fler detaljer när den körs.

6.2.2 Att exportera ett Linuxfilsystem via Appletalk.

Man kan exportera filsystem från sin Linuxburk till nätverket så att Applemaskinerna på nätverket kan använda dem.

För att göra detta så behöver man konfigurera filen `/usr/local/atalk/etc/AppleVolumes.system`. Det finns ytterligare en konfigurationsfil som heter `/usr/local/atalk/etc/AppleVolumes.default` som har precis samma format och beskriver vilka filsystem som kan användas av användare som ansluter med gäst-rättigheter.

Alla detaljer om hur man konfigurerar dessa finns i manualbladet för *afpd*.

Ett enkelt exempel:

```
/tmp Scratch
/home/ftp/pub "Public Area"
```

Detta skulle exportera `/tmp`-filsystemet som en AppleShare-volym 'Scratch' och `/home/ftp/pub`-katalogen som en AppleShare-volym 'Public Area'. Volymnamnen är inte obligatoriska, daemonen väljer namn om man inte anger dem, men det skadar inte att ange dem ändå.

6.2.3 Att dela sin skrivare via Appletalk.

Man kan dela sin Linuxskrivare med sina Applemaskiner ganska enkelt. Man behöver köra programmet *papd* (Appletalk Printer Access Protocol Daemon). När man kör detta program så tar det emot förfrågningar från Applemaskinerna och spoolar utskriftsjobben till den lokala line-printer daemonen.

Man behöver ändra filen `/usr/local/atalk/etc/papd.conf` för att konfigurera daemonen. Syntaxen för denna fil är densamma som för den vanliga `/etc/printcap` filen. Namnet som man ger till definitionen registreras med Appletalks namnsättningsprotokoll, NBP.

En exempelkonfiguration kan se ut så här:

```
TricWriter:\
   :pr=lp:op=cg:
```

Vilken skulle skapa en printer som kallas för 'TricWriter' som blir tillgänglig för Appletalknätverket och alla accepterade jobb skulle skrivas ut på Linuxskrivaren '`lp`' (som den definieras i filen `/etc/printcap`) genom att använda *lpd*. Uttrycket '`op=cg`' talar om att Linux-användaren '`cg`' är ansvarig för skrivaren.

6.2.4 Att starta programvaran för Appletalk.

Nu bör man var redo att testa denna enkla konfiguration. Det finns en fil *rc.atalk* som följer med paketet *netatalk* som borde fungera för de flesta, så allt man behöver göra är följande:

```
# /usr/local/atalk/etc/rc.atalk
```

och allt borde startas och fungera. Man skall inte se några felmeddelanden och programvaran kommer att skicka meddelanden till konsolen som indikerar varje steg som startas.

6.2.5 Att testa programvaran för Appletalk.

För att testa att mjukvaran fungerar som den skall, så går man till en av sina Applemaskiner, tar ner äpplemenyn, väljer Väljaren (Chooser), klickar på AppleShare, och Linuxburken bör synas.

6.2.6 Brister i programvaran för Appletalk.

- Man kanske måste starta Appletalk-stödet innan man konfigurerar sitt IP-nätverk. Om man har problem med att starta Appletalkprogrammen, eller om man, efter att ha startat dem, har problem med sitt IP-nätverk, så försöker man med att starta Appletalk innan man kör sin `/etc/rc.d/rc.inet1` fil.
- Daemonen *afpd* (Apple Filing Protocol Daemon) stökar till ordentligt i filsystemet. Under mountpunkterna så skapar den ett par kataloger som heter `.AppleDesktop` och `Network Trash Folder`. Sedan för varje katalog som man besöker så kommer den att skapa en `.AppleDouble` under dem så att den kan spara diverse information. Så man bör tänka efter innan man exporterar `/`, man kommer att ha det skoj när man städar upp efteråt.
- Daemonen *afpd* förväntar sig lösenord i klartext från Macarna. Säkerhet kan vara ett problem, så man skall vara försiktig när man kör denna daemonen på en dator som är ansluten till Internet, man har sig själv att skylla om någon elak person gör någon skada.
- Existerande diagnostiseringsverktyg som *netstat* och *ifconfig* stöder inte Appletalk. Rå information är tillgänglig i katalogen `/proc/net/` om man behöver den.

6.2.7 Mer information.

En mycket mer detaljerad information om hur man konfigurerar Appletalk för Linux finns i Anders Brownworths *Linux Netatalk-HOWTO* sida på *thehamptons.com* `<http://thehamptons.com/anders/netatalk/>`.

6.3 ATM

Werner Almesberger <`werner.almesberger@lrc.di.epfl.ch`> håller i ett projekt för att skapa stöd för Asynchronous Transfer Mode i Linux. Uppdaterad information om statusen för projektet kan fås från *lrcwww.epfl.ch* `<http://lrcwww.epfl.ch/linux-atm/>`.

6.4 AX25 (AF_AX25)

Enhetsnamn för AX.25 är 'sl0', 'sl1', osv i 2.0.* kärnor eller 'ax0', 'ax1', osv i 2.1.* kärnor.

Kompileringsalternativ för Kärnan:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
```

AX25, Netrom och Rose protokollen täcks av *AX25-HOWTO* <AX25-HOWTO.html>. Dessa protokoll används av Amatörradio-operatörer i hela världen för paketradio experiment.

Det mesta arbetet för att implementera protokollen har utförts av Jonathon Naylor, jsn@cs.nott.ac.uk.

6.5 DECNet

Stöd för DECNet håller för tillfället på att utvecklas. Man kan räkna med att det dyker upp i sena 2.1.* kärnor.

6.6 EQL - trafikutjämnare för multipla linor.

Enhetsnamnet för EQL är 'eql'. Med standardkärnan kan man endast ha en EQL-enhet per maskin. EQL tillhandahåller hjälpmedel för att använda multipla punkt till punkt förbindelser (tex PPP, SLIP eller PLIP) som en ensam logisk länk för att bära TCP/IP. Ofta är det billigare att använda flera linor med lägre hastighet än att ha en höghastighetslina installerad.

Kompileringsalternativ för Kärnan:

```
Network device support  --->
    [*] Network device support
    <*> EQL (serial line load balancing) support
```

För att stödja denna mekanism så måste maskinen på andra sidan av linorna också stödja EQL. Linux, Livingstone Portmasters och nyare dial-in servrar stöder kompatibla tjänster.

För att konfigurera EQL behöver man eql-verktygen som finns på: *sunsite.unc.edu* <ftp://sunsite.unc.edu/pub/linux/system/Serial/eql-1.2.tar.gz>.

Konfigurationen är hyfsat okomplicerad. Man börjar med att konfigurera eql-gränssnittet. Eql-gränssnittet är precis som alla andra nätverksgränssnitt. Man kan konfigurera IP-adressen och MTU genom att använda *ifconfig*, så ungefär som:

```
ifconfig eql 192.168.10.1 mtu 1006
```

Sedan behöver man manuellt initiera var och en av linorna som man skall använda. Dessa kan vara en valfri kombination av punkt till punkt förbindelser. Hur man initierar de anslutningarna beror på vilken typ av länkar de är, se passande sektioner för mer information.

Till sist skall man associera den seriella länken med EQL-enheten, detta kallas för 'enslaving' och görs med kommandot *eql_enslave*:

```
eql_enslave eql sl0 28800
eql_enslave eql ppp0 14400
```

Parametern '*estimated speed*' som man ger till *eql_enslave* gör ingen direkt nytta. Den används av EQL-drivrutinen för att avgöra hur stor del av datagrammen som den enheten skall få ta emot, så man kan finjustera balansen hos linorna genom att ändra på detta värde.

För att disassociera en lina från en EQL-enhet så använder man kommandot *eql_emancipate*:

```
eql_emancipate eql sl0
```

Man lägger till routing som om det vore en normal punkt till punkt förbindelse, förutom att router skall referera till `eql` enheten istället för de verkliga seriella enheterna:

```
route add default eql
```

EQL-drivrutinen utvecklades av Simon Janes, `simon@ncm.com`.

6.7 Ethernet

Enhetsnamn för Ethernet är '`eth0`', '`eth1`', '`eth2`' osv. Det första kortet som hittas får namnet '`eth0`' och resten tilldelas namn sekvensiellt i den ordning som de hittas.

För att ta reda på hur man får sitt Ethernet-kort att fungera i Linux så bör man titta i *Ethernet-HOWTO* `<Ethernet-HOWTO.html>`.

När man väl har en kärna som stöder sitt Ethernet-kort så är det enkelt att konfigurera kortet.

Vanligtvis så används ungefär följande:

```
# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
# route add -net 192.168.0.0 netmask 255.255.255.0 eth0
```

De flesta drivrutinerna för Ethernet utvecklades av Donald Becker, `becker@CESDIS.gsfc.nasa.gov`.

6.8 FDDI

Enhetsnamnen för FDDI är '`fddi0`', '`fddi1`', '`fddi2`' osv. Det första kortet som hittas får namnet '`fddi0`' och resten tilldelas namn sekvensiellt i den ordning som de hittas.

Larry Stefani, `lstefani@ultranet.com`, har utvecklat en drivrutin för EISA och PCI FDDI-kort från Digital Equipment Corporation.

Kompileringsalternativ för Kärnan:

```
Network device support  --->
    [*] FDDI driver support
    [*] Digital DEFEA and DEFPA adapter support
```

När man väl har en kärna som stöder sitt FDDI-kort, så konfigureras FDDI-kortet nästan likadant som Ethernet-kortet. Man behöver bara ange lämpliga FDDI-enhetsnamn till kommandona *ifconfig* och *route*.

6.9 Frame Relay

Enhetsnamnen för Frame Relay är '`dlci00`', '`dlci01`' osv för DLCI inkapslingsenheter och '`sdla0`', '`sdla1`' osv för FRAD(s).

Frame Relay är en ny teknik att bygga nätverk och är designad att passa datakommunikation vars trafik är av oregelbunden karaktär. Man ansluter till ett Frame Realay nätverk genom att använda en Frame Relay Access Device (FRAD). Linux Frame Relay stödjer IP över Frame Relay så som det beskrivs i RFC1490.

Kompileringsalternativ för Kärnan:

```
Network device support  --->
    <*> Frame relay DLCI support (EXPERIMENTAL)
    (24)   Max open DLCI
    (8)   Max DLCI per device
    <*>   SDLA (Sangoma S502/S508) support
```

Mike McLagan, `mike.mclagan@linux.org`, utvecklade stödet och konfigurationsverktygen för Frame Relay.

För närvarande är de FRADs som stöds följande: *Sangoma Technologies* `<http://www.sangoma.com/>` S502A, S502E och S508.

För att konfigurera FRAD och DLCI enheter efter det att man har kompilerat om sin kärna så behöver man konfigurationsverktygen för Frame Relay. Dessa finns på: *ftp.invlogic.com* `<ftp://ftp.invlogic.com/pub/linux/fr/frad-0.15.tgz>`. Det är okomplicerat att kompilera och installera verktygen, men avsaknaden av en toppnivå-Makefile gör det till en manuell process:

```
# cd /usr/src
# tar xvfz .../frad-0.15.tgz
# cd frad-0.15
# for i in common dlci frad; make -C $i clean; make -C $i; done
# mkdir /etc/frad
# install -m 644 -o root -g root bin/*.sfm /etc/frad
# install -m 700 -o root -g root frad/fradcfg /sbin
# install -m 700 -o root -g root dlci/dlcicfg /sbin
```

Efter att man har installerat verktygen skall man skapa en fil som heter `/etc/frad/router.conf`. Man kan använda följande mall, som är en modifierad version av en exempelfil:

```
# /etc/frad/router.conf
# This is a template configuration for frame relay.
# All tags are included. The default values are based on the code
# supplied with the DOS drivers for the Sangoma S502A card.
#
# A '#' anywhere in a line constitutes a comment
# Blanks are ignored (you can indent with tabs too)
# Unknown [] entries and unknown keys are ignored
#

[Devices]
Count=1                 # number of devices to configure
Dev_1=sdla0             # the name of a device
#Dev_2=sdla1            # the name of a device

# Specified here, these are applied to all devices and can be overriden for
# each individual board.
#
Access=CPE
Clock=Internal
KBaud=64
Flags=TX
```

```
#
# MTU=1500              # Maximum transmit IFrame length, default is 4096
# T391=10               # T391 value    5 - 30, default is 10
# T392=15               # T392 value    5 - 30, default is 15
# N391=6                # N391 value    1 - 255, default is 6
# N392=3                # N392 value    1 - 10, default is 3
# N393=4                # N393 value    1 - 10, default is 4

# Specified here, these set the defaults for all boards
# CIRfwd=16             # CIR forward   1 - 64
# Bc_fwd=16             # Bc forward    1 - 512
# Be_fwd=0              # Be forward    0 - 511
# CIRbak=16             # CIR backward  1 - 64
# Bc_bak=16             # Bc backward   1 - 512
# Be_bak=0              # Be backward   0 - 511


#
#
# Device specific configuration
#
#

#
# The first device is a Sangoma S502E
#
[sdla0]
Type=Sangoma            # Type of the device to configure, currently only
                        # SANGOMA is recognised
#
# These keys are specific to the 'Sangoma' type
#
# The type of Sangoma board - S502A, S502E, S508
Board=S502E
#
# The name of the test firmware for the Sangoma board
# Testware=/usr/src/frad-0.10/bin/sdla_tst.502
#
# The name of the FR firmware
# Firmware=/usr/src/frad-0.10/bin/frm_rel.502
#
Port=360                # Port for this particular card
Mem=C8                  # Address of memory window, A0-EE, depending on card
IRQ=5                   # IRQ number, do not supply for S502A
DLCIs=1                 # Number of DLCI's attached to this device
DLCI_1=16               # DLCI #1's number, 16 - 991
# DLCI_2=17
# DLCI_3=18
# DLCI_4=19
# DLCI_5=20
#
# Specified here, these apply to this device only,
# and override defaults from above
#
# Access=CPE            # CPE or NODE, default is CPE
```

```
# Flags=TXIgnore,RXIgnore,BufferFrames,DropAborted,Stats,MCI,AutoDLCI
# Clock=Internal        # External or Internal, default is Internal
# Baud=128              # Specified baud rate of attached CSU/DSU
# MTU=2048              # Maximum transmit IFrame length, default is 4096
# T391=10               # T391 value    5 - 30, default is 10
# T392=15               # T392 value    5 - 30, default is 15
# N391=6                # N391 value    1 - 255, default is 6
# N392=3                # N392 value    1 - 10, default is 3
# N393=4                # N393 value    1 - 10, default is 4

#
# The second device is some other card
#
# [sdla1]
# Type=FancyCard        # Type of the device to configure.
# Board=                # Type of Sangoma board
# Key=Value             # values specific to this type of device


#
# DLCI Default configuration parameters
# These may be overridden in the DLCI specific configurations
#
CIRfwd=64               # CIR forward   1 - 64
# Bc_fwd=16             # Bc forward    1 - 512
# Be_fwd=0              # Be forward    0 - 511
# CIRbak=16             # CIR backward  1 - 64
# Bc_bak=16             # Bc backward   1 - 512
# Be_bak=0              # Be backward   0 - 511

#
# DLCI Configuration
# These are all optional. The naming convention is
# [DLCI_D<devicenum>_<DLCI_Num>]
#

[DLCI_D1_16]
# IP=
# Net=
# Mask=
# Flags defined by Sangoma: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
# CIRfwd=64
# Bc_fwd=512
# Be_fwd=0
# CIRbak=64
# Bc_bak=512
# Be_bak=0

[DLCI_D2_16]
# IP=
# Net=
# Mask=
# Flags defined by Sangoma: TXIgnore,RXIgnore,BufferFrames
# DLCIFlags=TXIgnore,RXIgnore,BufferFrames
```

```
# CIRfwd=16
# Bc_fwd=16
# Be_fwd=0
# CIRbak=16
# Bc_bak=16
# Be_bak=0
```

När man har skapat sin `/etc/frad/router.conf` fil så är det enda som återstår att konfigurera enheterna. Detta är bara lite svårare än att konfigurera en normal nätverksenhet. Man måste komma ihåg att starta upp FRAD-enheten innan DLCI inkapslingsenheterna.

```
# Configure the frad hardware and the DLCI parameters
/sbin/fradcfg /etc/frad/router.conf || exit 1
/sbin/dlcicfg file /etc/frad/router.conf
#
# Bring up the FRAD device
ifconfig sdla0 up
#
# Configure the DLCI encapsulation interfaces and routing
ifconfig dlci00 192.168.10.1 pointopoint 192.168.10.2 up
route add -net 192.168.10.0 netmask 255.255.255.0 dlci00
#
ifconfig dlci01 192.168.11.1 pointopoint 192.168.11.2 up
route add -net 192.168.11.0 netmask 255.255.255.0 dlci00
#
route add default dev dlci00
#
```

6.10 IP-redovisning (IP Accounting)

Med IP-redovisningsegenskaperna i Linuxkärnan kan man samla ihop och analysera viss data från nätverksanvändningen. Datan som samlas ihop består av antalet paket och antalet bytes ackumulerade sedan talen senast nollställdes. Man kan specificera en mängd olika regler för att kategorisera talen för att passa sina ändamål.

Kompileringsalternativ för Kärnan:

```
Networking options  --->
    [*] IP: accounting
```

När man har kompilerat och installerat kärnan så behöver man använda kommandot *ipfwadm* för att konfigurera IP-redovisningen. Det finns många olika sätt att bryta ner redovisningsinformationen. Jag har valt ett enkelt exempel på vad som skulle kunna vara användbart, man kan läsa manualbladet för kommandot *ipfwadm* för mer information.

Scenario: man har ett Ethernet-nätverk som är anslutet till Internet via en PPP-länk. På sitt Ethernet har man en maskin som erbjuder ett antal tjänster och man är intresserad av att veta hur mycket trafik som genereras av telnet, rlogin, ftp och www trafik.

Man skulle då kunna använda följande:

```
#
# Flush the accounting rules
ipfwadm -A -f
#
```

```
# Add rules for local ethernet segment
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 20
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 20
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 23
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 23
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 80
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 80
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 513
ipfwadm -A out -a -P tcp -S 44.136.8.96/29 513
ipfwadm -A in -a -P tcp -D 44.136.8.96/29
ipfwadm -A out -a -P tcp -D 44.136.8.96/29
ipfwadm -A in -a -P udp -D 44.136.8.96/29
ipfwadm -A out -a -P udp  -D 44.136.8.96/29
ipfwadm -A in -a -P icmp -D 44.136.8.96/29
ipfwadm -A out -a -P icmp -D 44.136.8.96/29
#
# Rules for default
ipfwadm -A in -a -P tcp -D 0/0 20
ipfwadm -A out -a -P tcp -S 0/0 20
ipfwadm -A in -a -P tcp -D 0/0 23
ipfwadm -A out -a -P tcp -S 0/0 23
ipfwadm -A in -a -P tcp -D 0/0 80
ipfwadm -A out -a -P tcp -S 0/0 80
ipfwadm -A in -a -P tcp -D 0/0 513
ipfwadm -A out -a -P tcp -S 0/0 513
ipfwadm -A in -a -P tcp -D 0/0
ipfwadm -A out -a -P tcp -D 0/0
ipfwadm -A in -a -P udp -D 0/0
ipfwadm -A out -a -P udp  -D 0/0
ipfwadm -A in -a -P icmp -D 0/0
ipfwadm -A out -a -P icmp -D 0/0
#
# List the rules
ipfwadm -A -l -n
#
```

Det sista kommandot listar var och en av redovisningsreglerna och visar de ihopsamlade summorna.

En viktig notering är att när man analyserar datan är att **summan för alla regler som passar in kommer att ökas** så för att erhålla summor för enskilda protokoll så måste man räkna lite grann. Om jag tex ville veta hur mycket data som inte var ftp, telnet, rlogin eller www så skulle jag subtrahera de individuella summorna från den regel som passar in på alla portarna.

```
# ipfwadm -A -l -n
IP accounting rules
 pkts bytes dir prot source               destination          ports
    0     0 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 20
    0     0 out tcp  44.136.8.96/29       0.0.0.0/0            20 -> *
    0     0 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 23
    0     0 out tcp  44.136.8.96/29       0.0.0.0/0            23 -> *
   10  1166 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 80
   10   572 out tcp  44.136.8.96/29       0.0.0.0/0            80 -> *
  242  9777 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> 513
  220 18198 out tcp  44.136.8.96/29       0.0.0.0/0            513 -> *
  252 10943 in  tcp  0.0.0.0/0            44.136.8.96/29       * -> *
```

```
  231 18831 out tcp  0.0.0.0/0            44.136.8.96/29      * -> *
    0     0 in  udp  0.0.0.0/0            44.136.8.96/29      * -> *
    0     0 out udp  0.0.0.0/0            44.136.8.96/29      * -> *
    0     0 in  icmp 0.0.0.0/0            44.136.8.96/29      *
    0     0 out icmp 0.0.0.0/0            44.136.8.96/29      *
    0     0 in  tcp  0.0.0.0/0            0.0.0.0/0           * -> 20
    0     0 out tcp  0.0.0.0/0            0.0.0.0/0           20 -> *
    0     0 in  tcp  0.0.0.0/0            0.0.0.0/0           * -> 23
    0     0 out tcp  0.0.0.0/0            0.0.0.0/0           23 -> *
   10  1166 in  tcp  0.0.0.0/0            0.0.0.0/0           * -> 80
   10   572 out tcp  0.0.0.0/0            0.0.0.0/0           80 -> *
  243  9817 in  tcp  0.0.0.0/0            0.0.0.0/0           * -> 513
  221 18259 out tcp  0.0.0.0/0            0.0.0.0/0           513 -> *
  253 10983 in  tcp  0.0.0.0/0            0.0.0.0/0           * -> *
  231 18831 out tcp  0.0.0.0/0            0.0.0.0/0           * -> *
    0     0 in  udp  0.0.0.0/0            0.0.0.0/0           * -> *
    0     0 out udp  0.0.0.0/0            0.0.0.0/0           * -> *
    0     0 in  icmp 0.0.0.0/0            0.0.0.0/0           *
    0     0 out icmp 0.0.0.0/0            0.0.0.0/0           *
#
```

6.11 IP Aliasing

Det finns vissa applikationer där det är användbart att kunna tilldela flera IP-adresser till en och samma nätverksenhet. Titta i IP-Aliasing mini-HOWTO för mer information än vad man hittar här.

Kompileringsalternativ för Kärnan:

```
Networking options  --->
    ....
    [*] Network aliasing
    ....
    <*> IP: aliasing support
```

När man har kompilerat och installerat sin kärna med stöd för IP Aliasing så är det väldigt enkelt att konfigurera. Aliasen läggs till virtuella nätverksenheter som är associerade med den verkliga enheten. En enkel namnkonvention används, nämligen `<enhetsnamn>:<virtuellt enhetsnummer>`, tex `eth0:0`, `ppp0:1` osv. Notera att en virtuell enhet endast kan konfigureras *efter* det att den riktiga enheten har konfigurerats.

Till exempel, antag att man har ett Ethernet-nätverk som innehåller två olika IP-subnät på en gång. Man vill nu att maskinen skall ha direkt access till båda. Man skulle använda något som:

```
#
# ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# ifconfig eth0:0 192.168.10.1 netmask 255.255.255.0 up
# route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
#
```

För att ta bort ett alias så lägger man till ett '-' i slutet på dess namn:

```
# ifconfig eth0:0- 0
```

Alla router som är relaterade till det aliaset kommer också att tas bort automatiskt.

6.12 IP Brandväggar (IP Firewalls)

IP-brandväggar täcks mer i detalj i *Firewall-HOWTO* `<Firewall-HOWTO.html>`. Med en IP-brandvägg kan man skydda sin maskin mot otillåten åtkomst via nätverket genom att filtrera bort eller tillåta datagram till eller ifrån IP-adresser som man anger. Det finns regler i tre olika klasser: inkommande filtrering, utgående filtrering och filtrering vid vidareskickning. Reglerna för inkommande gäller datagram som tas emot av en nätverksenhet. Reglerna för utgående gäller datagram som skall skickas av en nätverksenhet. Reglerna för vidareskickning gäller de datagram som tas emot men inte skall till den aktuella maskinen, dvs datagram som skall routas.

Kompileringsalternativ för Kärnan:

```
Networking options  --->
    [*] Network firewalls
    ....
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: firewalling
    [ ] IP: firewall packet logging
```

Konfigurationen av reglerna för brandväggen görs med hjälp av kommandot *ipfwadm*. Som jag nämnde tidigare så är jag ingen expert på säkerhet, så även om jag visar ett exempel som går att använda så rekommenderas egna undersökningar i ämnet om säkerhet är särskilt viktigt.

Det kanske vanligaste användningsområdet för en brandvägg är när man använder sin Linuxburk som en router och brandvägg för att skydda sitt lokala nätverk mot otillåten åtkomst utifrån.

Följande konfiguration är baserad på ett bidrag från Arnt Gulbrandsen, <`agulbra@troll.no`>.

Exemplet beskriver en konfiguration av brandväggsreglerna i Linux-brandväggen/routern som illustreras i denna figur:

```
-                                   -
 \                                  | 172.16.37.0
  \                                 |   /255.255.255.0
   \                 ---------      |
    |  172.16.174.30 | Linux |      |
NET =================|  f/w  |------|    ..37.19
    |    PPP         | router|      |  --------
   /                 ---------      |--| Mail |
  /                                 |  | /DNS |
 /                                  |  --------
-                                   -
```

Kommandona som följer placeras normalt i en `rc` fil så att de startas automatiskt varje gång systemet startas. För maximal säkerhet så borde de utföras efter det att nätverksgränssnitten konfigurerats, men innan enheterna aktiveras så att man på detta sätt hindrar att någon får tillgång till maskinen medan den startar upp.

```
#!/bin/sh

# Flush the 'Forwarding' rules table
# Change the default policy to 'accept'
#
/sbin/ipfwadm -F -f
```

```
/sbin/ipfwadm -F -p accept
#
# .. and for 'Incoming'
#
/sbin/ipfwadm -I -f
/sbin/ipfwadm -I -p accept

# First off, seal off the PPP interface
# I'd love to use '-a deny' instead of '-a reject -y' but then it
# would be impossible to originate connections on that interface too.
# The -o causes all rejected datagrams to be logged. This trades
# disk space against knowledge of an attack of configuration error.
#
/sbin/ipfwadm -I -a reject -y -o -P tcp -S 0/0 -D 172.16.174.30

# Throw away certain kinds of obviously forged packets right away:
# Nothing should come from multicast/anycast/broadcast addresses
#
/sbin/ipfwadm -F -a deny -o -S 224.0/3 -D 172.16.37.0/24
#
# and nothing coming from the loopback network should ever be
# seen on a wire
#
/sbin/ipfwadm -F -a deny -o -S 127.0/8 -D 172.16.37.0/24

# accept incoming SMTP and DNS connections, but only
# to the Mail/Name Server
#
/sbin/ipfwadm -F -a accept -P tcp -S 0/0 -D 172.16.37.19 25 53
#
# DNS uses UDP as well as TCP, so allow that too
# for questions to our name server
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 -D 172.16.37.19 53
#
# but not "answers" coming to dangerous ports like NFS and
# Larry McVoy's NFS extension.  If you run squid, add its port here.
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 53 \
        -D 172.16.37.0/24 2049 2050

# answers to other user ports are okay
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 53 \
        -D 172.16.37.0/24 53 1024:65535

# Reject incoming connections to identd
# We use 'reject' here so that the connecting host is told
# straight away not to bother continuing, otherwise we'd experience
# delays while ident timed out.
#
/sbin/ipfwadm -F -a reject -o -P tcp -S 0/0 -D 172.16.37.0/24 113

# Accept some common service connections from the 192.168.64 and
# 192.168.65 networks, they are friends that we trust.
```

```
#
/sbin/ipfwadm -F -a accept -P tcp -S 192.168.64.0/23 \
        -D 172.16.37.0/24 20:23

# accept and pass through anything originating inside
#
/sbin/ipfwadm -F -a accept -P tcp -S 172.16.37.0/24 -D 0/0

# deny most other incoming TCP connections and log them
# (append 1:1023 if you have problems with ftp not working)
#
/sbin/ipfwadm -F -a deny -o -y -P tcp -S 0/0 -D 172.16.37.0/24

# ... for UDP too
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 -D 172.16.37.0/24
```

Att göra en bra konfiguration av brandväggen är lite trixigt. Exemplet ovan borde dock vara en hyfsad startpunkt. Manualbladet för kommandot *ipfwadm* innehåller mer hjälp om hur man använder det. Om man tänker konfigurera en brandvägg, så skall man se till att fråga runt och få så mycket råd man kan ifrån källor som man anser vara pålitliga. Låt sedan någon testa konfigurationen från utsidan.

6.13 IPIP Inkapsling (IPIP Encapsulation)

Varför skulle man vilja kapsla in IP-datagram i andra IP-datagram? Det låter som en konstig sak att göra om man aldrig sett ett exempel på det innan. Ok, här är ett par vanliga områden där det används: Mobil IP och IP-Multicast. Men där det antagligen används mest är också det mest okända området, Amatörradio.

Kompileringsalternativ för Kärnan:

```
Networking options  --->
    [*] TCP/IP networking
    [*] IP: forwarding/gatewaying
    ....
    <*> IP: tunneling
```

Tunnlingsenheterna heter '`tunl0`', '`tunl1`' osv.

"Men varför....?". Ok, ok. Konventionella regler för IP-routing säger att ett IP-nätverk består av en nätverksadress och en nätmask. Detta producerar en serie kontinuerliga adresser som alla kan routas till en och samma utgående lina. Detta är väldigt bekvämt, men det innebär att man bara kan använda en viss IP-adress när man är ansluten till ett visst nätverk som adressen tillhör. I de flesta fall går detta bra, men om man är en mobil 'nät-invånare' så kanske man inte alltid är ansluten till ett och samma nätverk hela tiden. IPIP inkapsling (IP-tunnling) gör det möjligt att slippa den restriktionen genom att låta datagram som är destinerade till en viss IP-adress packas in i ett nytt IP-paket och omdirigeras till en annan IP-adress. Om man vet att man kommer att arbeta på ett annat IP-nätverk ett tag så kan man ställa in en maskin på sitt hemmanät att ta emot ens datagram och sedan omdirigera dem till den adress som man använder temporärt.

6.13.1 En tunnlad nätverkskonfiguration.

Som alltid, så tycker jag att en figur fungerar bättre än en massa förvirrande text, så här kommer en:

```
 192.168.1/24                       192.168.2/24

     -                                    -
     |      ppp0 =            ppp0 =      |
     |  aaa.bbb.ccc.ddd  fff.ggg.hhh.iii  |
     |                                    |
     |   /-----\                /-----\   |
     |   |     |       //       |     |   |
     |---|  A  |------//---------|  B  |---|
     |   |     |     //         |     |   |
     |   \-----/                \-----/   |
     |                                    |
     -                                    -
```

Figuren illustrerar en annan möjlig anledning till IPIP-inkapsling, ett virtuellt privat nätverk. Detta exempel förutsätter att man har två maskiner som båda har en enkel uppringd anslutning till Internet. Båda datorerna allokeras en IP-adress. Bakom dessa maskiner finns några privata LAN konfigurerade med reserverade nätverksadresser. Antag att man vill låta vilken dator som helst på nätverk A kommunicera med vilken dator som helst på nätverk B, precis som om de vore anslutna till Internet med en nätverksroute. IPIP-inkapsling fixar detta. Notera att inkapslingen inte löser problemet med att låta datorerna på nätverken A och B kommunicera med någon annan dator på Internet, då behöver man trix som IP-maskering. Inkapsling görs normalt av maskiner som uppträder som routrar.

Linux routern `A` skulle konfigureras med:

```
#
PATH=/sbin:/usr/sbin
#
# Ethernet configuration
ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.1.1 up
route add -net 192.168.2.0 netmask 255.255.255.0 gw fff.ggg.hhh.iii tunl0
```

Linux routern `B` skulle konfigureras med:

```
#
PATH=/sbin:/usr/sbin
#
# Ethernet configuration
ifconfig eth0 192.168.2.1 netmask 255.255.255.0 up
route add -net 192.168.2.0 netmask 255.255.255.0 eth0
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.2.1 up
```

```
route add -net 192.168.1.0 netmask 255.255.255.0 gw aaa.bbb.ccc.ddd tunl0
```

Kommandot:

```
route add -net 192.168.1.0 netmask 255.255.255.0 gw aaa.bbb.ccc.ddd tunl0
```

betyder: 'Skicka alla datagram ämnade för `192.168.1.0/24` inuti ett IPIP-inkapslat datagram med destinationsadressen `aaa.bbb.ccc.ddd`'.

Notera att konfigurationen finns på båda sidor. Tunnlingsenheten använder '`gw`' parametern i routen som *destination* för IP-datagrammet som kapslar in originaldatagrammet. Den maskinen måste då veta hur man 'packar upp' ett IPIP-datagram, dvs den måste också ha en tunnlingsenhet.

6.13.2 En tunnlad datorkonfiguration.

Man måste inte routa ett helt nätverk. Man skulle till exempel kunna routa en enskild IP-adress. I sådana fall skulle man kunna konfigurera `tunl` enheten på den 'avlägsna' maskinen med sin hemma-IP-adress och i A-ändan bara använda dator-route (och Proxy Arp) istället för en nätverksroute via tunnlingsenheten. Låt oss rita om och modifiera vår konfiguration efter detta. Nu har vi bara datorn '`B`' som vill agera och uppföra sig som om den både var ansluten till Internet och även en del av nätverket som stöds av datorn '`A`':

```
 192.168.1/24

     -
     |      ppp0 =              ppp0 =
     |  aaa.bbb.ccc.ddd      fff.ggg.hhh.iii
     |
     |   /-----\                /-----\
     |   |     |       //       |     |
     |---|  A  |------//---------|  B  |
     |   |     |     //         |     |
     |   \-----/                \-----/
     |                      also: 192.168.1.12
     -
```

Linux routern '`A`' skulle konfigureras med:

```
#
PATH=/sbin:/usr/sbin
#
# Ethernet configuration
ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.1.1 up
route add -host 192.168.1.12 gw fff.ggg.hhh.iii tunl0
#
# Proxy ARP for the remote host
arp -s 192.168.1.12 xx:xx:xx:xx:xx:xx pub
```

Linuxdatorn ‘B’ skulle konfigureras med:

```
#
PATH=/sbin:/usr/sbin
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.1.12 up
route add -net 192.168.1.0 netmask 255.255.255.0 gw aaa.bbb.ccc.ddd tunl0
```

Denna typ av konfiguration är mer typisk för Mobil IP. Där en enskild dator vill flytta omkring på Internet och hela tiden använda samma IP-adress. Mer information om detta finns i sektionen om Mobil IP.

6.14 IPX (`AF_IPX`)

IPX-protokollet används mest i LAN-omgivningar med Novell NetWare(tm). Linux har stöd för att kunna agera som en nätverksändpunkt, eller som en router för IPX.

Kompileringsalternativ för Kärnan:

```
Networking options  --->
    [*] The IPX protocol
    [ ] Full internal IPX network
```

IPX-protokollet och NCPFS täcks mer detaljerat i *IPX-HOWTO* `<IPX-HOWTO.html>`.

6.15 IPv6

Precis när man tror att man förstår hur IP-nätverk fungerar så ändras reglerna! IPv6 är en förkortning av Internet Protocol version 6. IPv6 kan ibland också kallas för IPng (IP next generation) (SvÖ). IPv6 utvecklades i huvudsak för att ta bort oron i Internetvärlden att det snart är slut på IP-adresser som kan delas ut. IPv6-adresserna är 16 bytes stora (128 bits). IPv6 har ett antal andra ändringar, mest förenklingar, som kommer att göra IPv6-nätverk lättare att underhålla än IPv4-nätverk.

Linux har redan en fungerande, men inte komplett, IPv6-implementation i version `2.1.*` av kärnan.

Om man vill experimentera med nästa generations Internetteknik, eller om man behöver den, så bör man läsa IPv6-FAQ som finns på *www.terra.net* `<http://www.terra.net/ipv6/>`.

6.16 ISDN

Integrated Services Digital Network (ISDN) är en serie standarder som specificerar ett generellt switchat digitalt nätverk. En ISDN-‘uppringning’ skapar en synkron punkt till punkt förbindelse till destinationen. ISDN körs normalt på höghastighetslänkar som delas in i ett antal diskreta kanaler. Det finns två olika typer av kanaler, ‘B-kanaler’ som bär användardatan och ‘D-kanaler’ som används för kontrollinformation till ISDN-växeln. I Australien till exempel så kan ISDN levereras med en 2Mbps-länk som delas in i 30 diskreta 64kbps B-kanaler och en 64kbps D-kanal. Valfritt antal kanaler kan användas samtidigt i valfri kombination. Man kan till exempel upprätta 30 olika anslutningar på 64kbps var till 30 olika destinationer, eller så kan man upprätta 15 olika anslutningar på 128kbps var till 15 olika destinationer (med två kanaler per anslutning), eller bara ett litet antal kanaler och lämna resten vilande. En kanal kan användas till både inkommande och utgående anslutningar. Den ursprungliga avsikten med ISDN var att telefonbolagen ville kunna erbjuda en enda datatjänst som kunde ge antingen telefon- (digitalt) eller datatjänster till hem och kontor utan att kunden skulle behöva ändra någon konfiguration.

Det finns några olika sätt att ansluta en dator till en ISDN-tjänst. Ett sätt är att använda enhet som heter ‘Terminaladapter’ som ansluter till en ‘Network Terminating Unit’ vilken telefonbolaget har installerat i samband med ISDN-tjänsten och som har ett antal seriella gränssnitt. Ett av de gränssnitten används för att skicka kommandon som upprättar en förbindelse och konfiguration och de andra är anslutna till nätverket som skall användas. Linux fungerar i denna omgivningen utan modifikation, man behandlar bara terminaladaptern som vilken annan seriell enhet som helst. Ett annat sätt, vilket är så som kärnan stöder ISDN, är att installera ett ISDN-kort i Linuxburken och sedan låta mjukvaran i Linux hantera protokollen och upprätta förbindelser.

Kompileringsalternativ för Kärnan:

```
ISDN subsystem  --->
        <*> ISDN support
        [ ] Support synchronous PPP
        [ ] Support audio via ISDN
        < > ICN 2B and 4B support
        < > PCBIT-D support
        < > Teles/NICCY1016PC/Creatix support
```

Implementationen av ISDN i Linux stöder ett antal olika interna ISDN-kort. Dessa finns listade i konfigurationen för kärnan:

- ICN 2B and 4B
- Octal PCBIT-D
- Teles ISDN-cards and compatibles

Några av dessa kort kräver att man laddar hem särskild programvara för att de skall fungera. Det finns ett separat verktyg att göra detta med.

Alla detaljer om hur man konfigurerar ISDN för Linux finns tillgängligt i katalogen `/usr/src/linux/Documentation/isdn/` och en FAQ för *isdn4linux* finns på *www.lrz-muenchen.de* `<http://www.lrz-muenchen.de/~ui161ab/www/isdn/>`.

En anmärkning om PPP. PPP-protokollen fungerar antingen för asynkrona eller synkrona seriella linor. PPP-daemonen som vanligtvis levereras med Linux, ‘*pppd*’, stöder endast asynkront läge. Om man vill köra PPP-protokoll via ISDN så måste man använda en särskilt modifierad version. Var man hittar den finns beskrivet i dokumentationen som nämns ovan.

6.17 IP-maskering (IP Masquerade).

Många har en enkel uppringd anslutning till Internet. Nästan alla som använder den typen av konfiguration allokeras en enda IP-adress av ISPn. Det räcker normalt för att en dator skall ha full åtkomst till Internet. IP-maskering är ett smart trix som gör det möjligt att ha många maskiner som använder en enda IP-adress genom att låta de andra datorerna se ut som, därav termen maskering, maskinen som har den uppringda anslutningen. Det finns dock en liten brist, maskeringsfunktionen fungerar nästan alltid bara i ena riktningen. Det betyder att de förklädda datorerna kan ansluta utåt, men de kan inte ta emot nätverksanslutningar från andra datorer. Detta betyder att vissa nätverkstjänster inte fungerar, tex *talk*, och andra som till exempel *ftp* måste konfigureras att köra i passivt (PASV) läge för att fungera. Lyckligtvis så fungerar de vanligaste tjänsterna som till exempel *telnet*, *World Wide Web* och *irc* utmärkt.

Kompileringsalternativ för Kärnan:

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
    [*] Network firewalls
    ....
    [*] TCP/IP networking
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: masquerading (EXPERIMENTAL)
```

Vanligtvis har man sin Linuxburk konfigurerad för uppringd SLIP eller PPP, precis som om det var en fristående dator. Dessutom så skulle den ha en ytterligare nätverksenhet konfigurerad, kanske Ethernet som är konfigurerat med en reserverad nätverksadress. Datorerna som skall förklädas skulle då finnas på det nätverket. Var och en av de datorerna skulle ställa in IP-adressen på Linuxboxens Ethernet-kort som 'default gateway' eller router. Observera att de andra datorerna (de som inte har den uppringda anslutningen) inte behöver konfigureras med IP-maskering, de behöver endast veta vilken dator som är 'default gateway' (SvÖ).

En typisk konfiguration:

```
-                                  -
 \                                 | 192.168.1.0
  \                                |   /255.255.255.0
   \                 ---------     |
    |                | Linux | .1.1 |
NET =================| masq  |------|
    |    PPP/slip    | router|      |  --------
   /                 ---------      |--| host |
  /                                 |  |      |
 /                                  |  --------
-                                  -
```

De viktigaste kommandona för denna konfiguration är:

```
# Network route for ethernet
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# Default route to the rest of the internet.
route add default ppp0
#
# Cause all hosts on the 192.168.1/24 network to be masqueraded.
ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0
```

Man kan hitta mer information om detta på *IP Masquerade Resource Page* <http://www.hwy401.com/achau/ipmasq/> eller i IP-Masquerade-MINI-HOWTO.

6.18 IP Transparent Proxy

Med IP transparent proxy kan man omdirigera servrar eller tjänster ämnade för en annan dator till tjänsterna på denna maskinen. Detta kan till exempel vara användbart om man har en Linuxburk som router och även tillhandahåller en proxy-server. Då skulle man omdirigera alla anslutningar ämnade för en avlägsen tjänst till den lokala proxy-servern.

Kompileringsalternativ för Kärnan:

```
Code maturity level options  --->
        [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
        [*] Network firewalls
        ....
        [*] TCP/IP networking
        ....
        [*] IP: firewalling
        ....
        [*] IP: transparent proxy support (EXPERIMENTAL)
```

Man konfigurerar transparent proxy med kommandot *ipfwadm*.

Ett exempel som kan vara användbart följer:

```
ipfwadm -I -a accept -D 0/0 telnet -r 2323
```

Detta exempel gör att alla försök att upprätta en `telnet`-anslutning (port 23) till någon dator kommer att omdirigeras till port 2323 på denna dator. Om man kör en tjänst på den porten så kan man vidarebefodra telnet-anslutningar, logga dem eller göra vadhelst man önskar.

Ett mer intressant exempel är att omdirigera all `http`-trafik genom en lokal cache. Men protokollet som används av proxy-servrar är annorlunda än vanlig http: där en klient ansluter till `www.server.com:80` och frågar efter `/sökväg/sida/`, men när den ansluter till en lokal cache kontaktar den `proxy.local.domain:8080` och frågar efter `www.server.com/path/page`.

För att filtrera en `http`-förfrågan genom den lokala proxyn så behöver man lägga till det protokollet genom att köra en liten server, som heter `transproxy` (som man hittar på www). Man kan då köra `transproxy` på port 8081 och ge följande kommando:

```
ipfwadm -I -a accept -D 0/0 80 -r 8081
```

Programmet *transproxy* kommer då att ta emot alla anslutningar ämnade för externa servrar och dirigera dem till den lokala proxyn efter att den har fixat till protokollskillnaderna.

6.19 Mobil IP

Termen IP-mobilitet beskriver förmågan att för en dator flytta sina nätverksanslutningar från en punkt på Internet till en annan utan att ändra sin IP-adress eller tappa anslutningen. Vanligtvis när en IP-dator byter anslutningspunkt så måste den även byta IP-adress. IP-mobilitet övervinner detta problem genom att allokera en fix IP-adress till den mobila datorn och använda IP-inkapsling (tunnling) med automatisk routing för att se till att datagrammen som är ämnade för den routas till den IP-adress som den verkligen använder.

Ett projekt är igång för att ta fram ett komplett paket med verktyg för IP-mobilitet i Linux. Information om projektet kan fås på *Linux Mobile IP Home Page* `<http://anchor.cs.binghamton.edu/~mobileip/>`.

6.20 Multicast

Med IP-multicasting kan datagram routas till ett godtyckligt antal IP-datorer på olika IP-nätverk samtidigt. Denna mekanism kan till exempel användas för att sända broadcastmaterial, som tex video eller ljud, till ett stort antal datorer på Internet samtidigt utan att var och en av dessa datorer behöver belasta nätverket med en egen 'kopia' av materialet.

Kompileringsalternativ för Kärnan:

```
Networking options  --->
        [*] TCP/IP networking
        ....
        [*] IP: multicasting
```

Man behöver även ett paket med verktyg och göra mindre nätverkskonfiguration. Ett ställe att hämta information om hur man installerar dessa för Linux finns på: *www.teksouth.com* <http://www.teksouth.com/linux/multicast/>.

6.21 NAT - Översättning av nätverksadresser (Network Address Translation)

NAT är en standardiserad storebror till Linux IP-maskering. Det finns en detaljerad specifikation i RFC1631. NAT tillhandahåller funktioner som IP-maskering inte gör vilket gör det mer passande för användning i brandväggsroutrar hos företag och i större installationer.

En alpha-implementation av NAT för Linux 2.0.29 kärnan har utvecklats av Michael Hasenstein, Michael.Hasenstein@informatik.tu-chemnitz.de. Michaels dokumentation och implementation finn på *Linux IP Network Address Web Page* <http://www.csn.tu-chemnitz.de/HyperNews/get/linux-ip-nat.html>

Nyare Linux 2.1.* kärnor har också viss NAT-funktionalitet i routingalgoritmen.

6.22 NetRom (AF_NETROM)

Enhetsnamn för NetRom är 'nr0', 'nr1', osv.

Kompileringsalternativ för Kärnan:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    [*] Amateur Radio NET/ROM
```

Protokollen AX25, Netrom och Rose finns beskrivna i *AX25-HOWTO* <AX25-HOWTO.html>. Dessa protokoll används av radioamatörer i hela världen i experiment med paketradio.

Det mesta arbetet med implementationen av dessa protokoll har gjorts av Jonathon Naylor, jsn@cs.nott.ac.uk.

6.23 PLIP (Parallel Line Internet Protocol)

Enhetsnamn för PLIP är 'plip0', 'plip1 and plip2.

Kompileringsalternativ för Kärnan:

```
Networking options  --->
    <*> PLIP (parallel port) support
```

PLIP, är som SLIP i den mening att det används för att skapa en *punkt till punkt* nätverksförbindelse mellan två maskiner. Men det skiljer sig genom att det är designat för att använda de parallella skrivarportarna på datorn istället för de seriella (ett kabelschema finns längre fram i dokumentet). Eftersom det är möjligt att överföra mer än en bit åt gången med en parallellport, så är det möjligt att uppnå högre hastigheter med *PLIP*-gränssnittet än vad man gör med seriell enhet. Dessutom kan även den enklaste av alla parallellportar, skrivarporten, användas i stället för att man skall behöva köpa jämförelsevis dyra 16550AFN UARTs till de seriella portarna. PLIP använder dock mycket CPU-tid jämfört med en seriell länk och är naturligtvis inget bra val om man kan få tag på några billiga Ethernet-kort, men det fungerar om inget annat finns tillgängligt och det fungerar dessutom ganska bra. Man kan förvänta sig en överföringshastighet på ungefär 20 kB/s när en länk fungerar bra.

PLIP-drivrutinerna slåss med parallell-drivrutinen om hårdvaran. Om man vill använda båda drivrutinerna så skall man kompilera båda som moduler så kan man välja vilken port man skall använda för PLIP och vilka portar man skall använda för skrivardrivrutinen. Se *Modules-HOWTO* `<Modules-HOWTO.html>` för mer information om hur man konfigurerar moduler till kärnan.

Notera att vissa bärbara datorer använder chipsets som inte fungerar med PLIP därför att de inte tillåter vissa kombinationer av signaler som PLIP behöver, som skrivare inte använder.

Linux *PLIP*-gränssnitt är kompatibelt med *Crynwyr Packet Driver PLIP* vilket betyder att man kan ansluta sin Linuxburk till en DOS-maskin som kör en annan typ av TCP/IP via *PLIP*.

I 2.0.* kärnor är PLIP-enheterna mappade mot I/O-port och IRQ som följer:

```
device  i/o     IRQ
------  -----   ---
plip0   0x3bc   5
plip1   0x378   7
plip2   0x278   2
```

Om man inte har parallellportar som stämmer överens med någon av ovanstående kombinationer så kan man ändra en ports IRQ med kommandot *ifconfig* och parametern '`irq`'. Man måste då slå på IRQ på skrivarportarna i sitt ROM BIOS (om det stöder det).

I senare 2.1.* kärnor med Plug'n'Play stöd så allokeras PLIP-enheterna sekvensiellt när de hittas precis som Ethernet-enheterna.

När man kompilerar kärnan är det en fil som man kanske behöver titta i för att konfigurera *PLIP*. Filen är `/usr/src/linux/driver/net/CONFIG` och den innehåller *PLIP* timrar i millisekunder. De förvalda är antagligen ok i de flesta fall. Man måste antagligen öka på dem om man har en särskilt långsam dator, då man faktiskt skall öka timrarna på den *andra* datorn. Det finns ett program *plipconfig* med vilket man kan ändra dessa timerinställningar utan att kompilera om kärnan. Det kommandot följer med i många Linuxdistributioner.

För att konfigurera ett *PLIP*-gränssnitt så måste man **lägga till** följande rader i sin `rc`-fil för nätverket:

```
#
# Attach a PLIP interface
#
#  configure first parallel port as a plip device
/sbin/ifconfig plip0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# End plip
```

Där:

IPA.IPA.IPA.IPA

representerar ens egen IP-adress.

IPR.IPR.IPR.IPR

representerar IP-adress på den andra maskinen.

Parametern *pointopoint* har samma betydelse som för SLIP, den specificerar adressen på maskinen i andra änden av länken.

I nästan alla fall kan man behandla *PLIP*-gränssnittet som om det var ett *SLIP*-gränssnitt, förutom att varken *dip* eller *slattach* behöver, eller kan användas.

Mer information om PLIP kan hittas i: *PLIP-mini-HOWTO* `<mini/PLIP>`

6.24 PPP (Point to Point Protocol)

Enhetsnamn för PPP är 'ppp0', 'ppp1, osv. Enheter numreras sekvensiellt och den första enheten som konfigureras får 'ppp0'.

Kompileringsalternativ för Kärnan:

```
Networking options  --->
    <*> PPP (point-to-point) support
```

Detaljer om PPP-konfiguration finns i *PPP-HOWTO* `<PPP-HOWTO.html>`.

6.24.1 Att vidhålla en permanent anslutning till nätet med *pppd*.

Om man har en semi-permanent anslutning till nätet och vill att ens maskin automatiskt skall återuppta PPP-anslutningen om den bryts så finns det ett enkelt trix som gör detta:

Konfigurera PPP så att det kan startas genom att `root`-användaren ger kommandot:

```
# pppd
```

Se till att '`-detach`' parametern finns konfigurerad i filen `/etc/ppp/options`. Sedan skall följande rader läggas in i filen `/etc/inittab`, tillsammans med *getty*-definitionerna:

```
pd:23:respawn:/usr/sbin/pppd
```

Detta gör så att programmet *init* håller koll på *pppd* och automatiskt startar om det ifall det dör.

6.25 Rose protokollet (AF_ROSE)

Enhetsnamn för Rose är '`rs0`', '`rs1`', osv. i `2.1.*` kärnor. Rose finns endast i `2.1.*` kärnor.

Kompileringsalternativ för Kärnan:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    [*] Amateur Radio NET/ROM
```

Protokollen AX25, Netrom och Rose finns beskrivna i *AX25-HOWTO* `<AX25-HOWTO.html>`. Dessa protokoll används av radioamatörer i hela världen i experiment med paketradio.

Det mesta arbetet med implementationen av dessa protokoll har gjorts av Jonathon Naylor, `jsn@cs.nott.ac.uk`.

6.26 SAMBA (stöd för 'NetBEUI', 'NetBios').

SAMBA är en implementation av protokollet 'Session Management Block'. Med SAMBA kan system från bland annat Microsoft använda diskar och skrivare i en Linuxbox.

Detaljer om SAMBA och dess konfiguration finns i *SMB-HOWTO* <SMB-HOWTO.html>.

6.27 SLIP (Serial Line Internet Protocol) klient.

Enhetsnamn för SLIP är 'sl0', 'sl1' osv. Enheter numreras sekvensiellt och den första enheten som konfigureras får 'sl0'.

Kompileringsalternativ för Kärnan:

```
Network device support  --->
    [*] Network device support
    <*> SLIP (serial line) support
    [ ]  CSLIP compressed headers
    [ ]  Keepalive and linefill
    [ ]  Six bit SLIP encapsulation
```

Med SLIP kan man använda TCP/IP över en seriell lina, som kan vara en telefonledning och modem, eller en hyrd ledning av något slag. För att kunna använda SLIP behöver man ha tillgång till en *SLIP-server* i sitt närområde. Många universitet och företag i hela världen tillhandahåller SLIP-åtkomst.

SLIP använder de seriella portarna på datorn för att bära IP-datagram. För att göra detta måste det ha kontroll över de seriella enheterna. SLIP-enheter benämns med *sl0*, *sl1* osv. Hur motsvarar detta de seriella enheterna? Nätverkskoden använder vad som heter ett *ioctl*-anrop (i/o control) för att ändra de seriella enheterna till SLIP-enheter. Det finns två program som kan göra detta, de heter *dip* och *slattach*.

6.27.1 dip (Dialup IP)

dip är ett smart program som kan ställa in hastigheten på den seriella enheten, kommendera modemet att ringa upp den andra ändan av länken, automatiskt logga in på servern, söka efter meddelanden som man får från servern och ta fram information från dem som tex IP-adress och dessutom kan programmet utföra *ioctl*-kommandot som behövs för att sätta den seriella porten i SLIP-läge. *dip* har ett kraftfullt scriptspråk i vilket man kan automatisera sitt inloggningsförfarande.

Programmet finns på: *sunsite.unc.edu* <ftp://sunsite.unc.edu/pub/Linux/system/Network/serial/dip/dip337o-uri.tgz>.

För att installera det, försök med följande:

```
#
# cd /usr/src
# gzip -dc dip337o-uri.tgz | tar xvf -
# cd dip-3.3.7o

<edit Makefile>

# make install
#
```

`Makefile` antar att det finns en grupp som heter *uucp*, men om man vill kan man ändra det till *dip* eller *SLIP* beroende på sin konfiguration.

6.27.2 slattach

I motsats till *dip* så är *slattach* ett väldigt enkelt program som är väldigt enkelt att använda, men det är inte så sofistikerat som *dip*. Det har inte något scriptspråk och allt det gör är att konfigurera den seriella enheten som en SLIP-enhet. Det antar att man har all information man behöver och att den seriella länken är upprättad när man kör programmet. *slattach* är idealiskt att använda om man har en permanent förbindelse till en server, till exempel en fysisk kabel eller en hyrd ledning.

6.27.3 När använder man vilket?

Man skulle använda *dip* om man har en uppringd förbindelse, eller någon annan temporär förbindelse, till sin SLIP-server. Man skulle använda *slattach* om man har en hyrd ledning mellan sin maskin och SLIP-servern och när man inte behöver göra något särskilt för att förbindelsen skall fungera. Se avsnittet 'Permanent SLIP-anslutning' för mer information.

Att konfigurera SLIP är ungefär som att konfigurera ett Ethernet-gränssnitt (Läs avsnittet 'Att konfigurera en Ethernet-enhet' ovan). Det finns dock vissa viktiga skillnader.

Först och främst så är SLIP-länkar olika Ethernet-nätverk i den meningen att det alltid endast finns två datorer på nätverket, en i varje ända av länken. Till skillnad från Ethernet, som är tillgängligt så fort man är inkopplad, så måste man kanske, beroende på typen av länk, initialisera anslutningen på något speciellt sätt.

Om man använder *dip* så gör man det normalt inte vid systemstarten, utan senare när man är redo att använda länken. Det är möjligt att automatisera den processen. Om man använder *slattach* så vill man antagligen lägga till några rader i sin *rc.inet1*-fil. Detta beskrivs snart.

Det finns två huvudtyper av SLIP-servrar: Dynamisk IP-adress servrar och statisk IP-adress servrar. Nästan alla SLIP-servrar presenterar en prompt där man skall logga in med användarnamn och lösenord. *dip* kan logga in automatiskt.

6.27.4 Statisk SLIP-server med uppringd förbindelse och *dip*

En *statisk* SLIP-server är en server i vilken man har fått en IP-adress som är ens egen. Varje gång man ansluter till servern så konfigurerar man sin SLIP-port med den adressen. Den statiska SLIP-servern kommer att svara på modemuppringningen, eventuellt fråga efter användarnamn och lösenord, och sedan routa alla datagram ämnade för ens IP-adress genom den anslutningen. Om man har en statisk server, så kanske man vill lägga in rader med sitt datornamn och sin IP-adress (eftersom man vet vad den kommer att vara) i sin fil `/etc/hosts/`. Man bör också konfigurera lite andra filer, såsom `rc.inet2`, `host.conf`, `resolv.conf`, `/etc/HOSTNAME` och `rc.local`. Kom ihåg att när man konfigurerar `rc.inet1` så behöver man inte lägga till några särskilda kommandon för SLIP eftersom *dip* sköter om allt som behöver göras där. Man behöver dock ge *dip* all nödvändig information så att det kan konfigurera gränssnittet efter det att det har upprättat förbindelsen och loggat in på SLIP-servern.

Om det är såhär ens SLIP-server fungerar så kan man gå vidare till avsnittet 'Att använda *dip*' för att få reda på hur man konfigurerar *dip*.

6.27.5 Dynamisk SLIP-server med uppringd förbindelse och *dip*.

En *dynamisk* SLIP-server är en server där man slumpvis allokeras en IP-adress, från en pool med adresser, varje gång man loggar på. Detta betyder att det inte finns någon garanti för att man har en viss adress varje gång och att adressen kan användas av någon annan efter det att man har loggat av. Administratören för SLIP-servern har angett en mängd IP-adresser där servern väljer den första lediga när den får en ny anslutning varefter den guidar användaren igenom loginprocessen och sedan skriver ett välkomstmeddelande som innehåller IP-adressen som sedan används under resten av anslutningen.

Konfigurationen för denna typ av server liknar den för en statisk server förutom att man måste lägga till ett steg där man tar emot IP-adressen som servern har allokerat och konfigurera SLIP-enheten med den.

Återigen så gör *dip* det hårda jobbet och nyare versioner är tillräckligt smarta för att inte bara logga in, utan även ta reda på IP-adressen och spara den så att man kan konfigurera SLIP-enheten med den.

Om det är såhär ens SLIP-server fungerar så kan man gå vidare till avsnittet 'Att använda *dip*' för att få reda på hur man konfigurerar *dip*.

6.27.6 Att använda *dip*.

Som nämnts tidigare så är *dip* ett kraftfullt program som kan förenkla och automatisera processen där man ringer upp SLIP-servern, loggar in, aktivera anslutningen och konfigurera SLIP-enheter med lämpliga *ifconfig* och *route* kommandon.

För att använda *dip* så skall man skriva ett 'dip script', som formellt är en lista av kommandon som *dip* förstår och som talar om för *dip* hur det skall utföra varje sak som man vill att det skall utföra. Se `sample.dip` för att få en aning om hur det fungerar. *dip* är ett ganska kraftfullt program med många alternativ. Istället för att gå in på alla här så är det lämpligt att titta på manualbladet, README- och exempelfiler för *dip*.

Exempelfilen `sample.dip` förutsätter att man använder en statisk SLIP-server, där man vet sin IP-adress i förväg. För dynamiska SLIP-servrar så har nyare versioner av *dip* ett kommando som automatiskt läser och konfigurerar SLIP-enheten med IP-adressen som man allokeras. Följande exempel är en modifierad version av `sample.dip` som kom med *dip337j-uri.tgz* och kan vara en bra utgångspunkt. Det kan vara bra att spara det som `/etc/dipscript` och sedan ändra det efter behov:

```
#
# sample.dip    Dialup IP connection support program.
#
#               This file (should show) shows how to use the DIP
#       This file should work for Annex type dynamic servers, if you
#       use a static address server then use the sample.dip file that
#       comes as part of the dip337-uri.tgz package.
#
#
# Version:      @(#)sample.dip  1.40    07/20/93
#
# Author:       Fred N. van Kempen, <waltje@uWalt.NL.Mugnet.ORG>
#

main:
# Next, set up the other side's name and address.
# My dialin machine is called 'xs4all.hacktic.nl' (== 193.78.33.42)
get $remote xs4all.hacktic.nl
# Set netmask on sl0 to 255.255.255.0
netmask 255.255.255.0
# Set the desired serial port and speed.
port cua02
speed 38400

# Reset the modem and terminal line.
# This seems to cause trouble for some people!
reset
```

```
# Note! "Standard" pre-defined "errlevel" values:
#  0 - OK
#  1 - CONNECT
#  2 - ERROR
#
# You can change those grep'ping for "addchat()" in *.c...

# Prepare for dialing.
send ATQ0V1E1X4\r
wait OK 2
if $errlvl != 0 goto modem_trouble
dial 555-1234567
if $errlvl != 1 goto modem_trouble

# We are connected.  Login to the system.
login:
sleep 2
wait ogin: 20
if $errlvl != 0 goto login_trouble
send MYLOGIN\n
wait ord: 20
if $errlvl != 0 goto password_error
send MYPASSWD\n
loggedin:

# We are now logged in.
wait SOMEPROMPT 30
if $errlvl != 0 goto prompt_error

# Command the server into SLIP mode
send SLIP\n
wait SLIP 30
if $errlvl != 0 goto prompt_error

# Get and Set your IP address from the server.
#   Here we assume that after commanding the SLIP server into SLIP
#   mode that it prints your IP address
get $locip remote 30
if $errlvl != 0 goto prompt_error

# Set up the SLIP operating parameters.
get $mtu 296
# Ensure "route add -net default xs4all.hacktic.nl" will be done
default

# Say hello and fire up!
done:
print CONNECTED $locip ---> $rmtip
mode CSLIP
goto exit

prompt_error:
print TIME-OUT waiting for sliplogin to fire up...
goto error
```

```
login_trouble:
print Trouble waiting for the Login: prompt...
goto error

password:error:
print Trouble waiting for the Password: prompt...
goto error

modem_trouble:
print Trouble occurred with the modem...
error:
print CONNECT FAILED to $remote
quit

exit:
exit
```

Ovanstående exempel förutsätter att man kontaktar en *dynamisk* SLIP-server, om man kontaktar en *statisk* server så bör `sample.dip` som följer med *dip337-uri.tgz* fungera bra.

Där *dip* får kommandot *get $local* så söker det igenom den inkommande texten efter en textsträng som ser ut som en IP-adress, dvs tal som är separerade med '.'-tecken. Denna modifiering gjordes speciellt för att fungera med *dynamiska* SLIP-servrar, så att processen att läsa in IP-adressen från servern skulle bli automatiserad.

Ovanstående exempel skapar automatiskt en 'default route' via SLIP-länken, om det inte är detta man vill, om man tex har en Ethernet-anslutning som skall vara 'default route, så tar man bort kommandot *default* från scriptet.

Om man, då scriptet kört färdigt, ger kommandot *ifconfig* så ser man att det finns en enhet *sl0*. Detta är SLIP-enheten. Om så behövs så kan man modifiera dess konfiguration manuellt efter det att *dip* har kört färdigt, genom att använda kommandona *ifconfig* och *route*.

Notera att med *dip* kan man välja ett antal olika protokoll att använda med `mode`-kommandot, det vanligaste är *cSLIP* för SLIP med kompression. Notera att båda ändar av länken måste komma överens, så man måste välja det som ens server är inställt på.

Ovanstående exempel är hyfsat stabilt och skall klara av de flesta felen. Titta annars i manualbladet för *dip* för mer information. Naturligtvis så kan man skriva scriptet så att det till exempel försöker ringa upp servern igen om det inte lyckas få en anslutning inom en given tid, eller till och med försöka med en serie olika servrar om man har tillgång till mer än en.

6.27.7 Permanent SLIP-anslutning med hyrd ledning och *slattach*.

Om man har en kabel mellan två maskiner, eller är lyckligt lottad och har en hyrd ledning, eller någon annan permanent seriell anslutning mellan sin maskin och en annan, så behöver man inte besvära sig med att använda *dip* för att sätta upp en seriell länk. *slattach* är ett väldigt enkelt verktyg att använda som har precis tillräckligt med funktionalitet för att konfigurera en anslutning.

Eftersom anslutningen är permanent så vill man lägga till några kommandon i sin `rc.inet1`-fil. Det enda man egentligen behöver göra för en permanent anslutning är att konfigurera den seriella enheten till korrekt hastighet och ställa in den i SLIP-läge. Med *slattach* kan man göra detta i ett enda kommando. Man **lägger till** följande i sin `rc.inet1`-fil:

```
#
```

```
# Attach a leased line static SLIP connection
#
#  configure /dev/cua0 for 19.2kbps and cslip
/sbin/slattach -p cslip -s 19200 /dev/cua0 &
/sbin/ifconfig sl0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# End static SLIP.
```

Där:

IPA.IPA.IPA.IPA

representerar IP-adressen.

IPR.IPR.IPR.IPR

representerar the IP-adressen i andra ändan.

slattach allokerar den första icke allokerade SLIP-enheten till den specificerade seriella enheten. *slattach* börjar börjar med *sl0*. Därför så parar *slattach* ihop *sl0* med den specifierade seriella enheten och därefter *sl1* osv.

Med *slattach* kan man konfigurera ett antal olika protokoll med parametern `-p`. Vanligtvis så använder man *SLIP* eller *cSLIP* beroende på om man vill ha kompression eller inte, men båda sidor måste vara överens.

6.28 SLIP (Serial Line Internet Protocol) server.

Om man har en maskin, som kanske är nätverksansluten, som man vill att andra skall kunna ansluta till och använda nätverkstjänster, så skall man konfigurera sin maskin som en server. Om man vill använda SLIP som protokoll så har man för närvarande tre valmöjligheter för hur man skall konfigurera Linuxboxen som en SLIP-server. Mitt råd skulle vara att använda det som presenteras först , *sliplogin*, eftersom det verkar vara det enklaste att konfigurera och förstå, men jag kommer att presentera en sammanfattning av varje så att man kan bilda sig en egen uppfattning.

6.28.1 SLIP-server med *sliplogin*.

sliplogin är ett program som man kan använda istället för det normala login-shellet för SLIP-användare som konverterar terminallinjen till en SLIP-länk. Man kan med *sliplogin* konfigurera sin Linuxbox som antingen en *statisk adressserver*, användare får samma IP-adress varje gång de ansluter, eller som en *dynamisk adresserver* där användare inte nödvändigtvis får samma IP-adress varje gång de ansluter.

De som ansluter kommer att logga in på vanligt vis med användarnamn och lösenord, men istället för att få ett shell efter de loggat in så exekveras *sliplogin*, som söker i sin konfigurationsfil (`/etc/slip.hosts`) efter en rad med ett loginnamn som motsvarar det som använts vid inloggningen. Om detta hittas så konfigurerar *sliplogin* linjen som en ren 8 bitars länk och använder *ioctl* för att konvertera länken till en SLIP-länk. Sedan återstår ett steg där *sliplogin* kör ett shellscript som konfigurerar SLIP-gränssnittet med relevant IP-adress, nätmask och routing. Scriptet heter normalt `/etc/slip.login`, men på liknande sätt som för *getty* om man har särskilda användare som behöver speciell initiering, så kan man skapa script som heter `/etc/slip.login.loginname` som körs istället för det allmänna.

Det finns antingen tre eller fyra filer att konfigurera för att *sliplogin* skall fungera. Jag skall i detalj gå igenom hur man får tag på programvara och hur man konfigurerar detta. Filerna är:

- `/etc/passwd`, för användarkonton.

- `/etc/slip.hosts`, för att innehålla information som är unik för varje användare.
- `/etc/slip.login`, vilken tar hand om konfiguration av routing.
- `/etc/slip.tty`, vilken endast behövs om man skall konfigurera en server som skall använda *dynamisk adressallokering* och innehåller en tabell med adresser att allokera.
- `/etc/slip.logout`, vilken innehåller kommandon för att städa upp efter det att en användare har lagt på eller loggat ut.

Var man får tag på *sliplogin*. Vissa kanske redan har *sliplogin*-paketet installerat som en del av sin distribution, om inte så kan *sliplogin* hämtas ifrån: *sunsite.unc.edu* `<ftp://sunsite.unc.edu/pub/linux/system/Network/serial/sliplogin-2.1.1.tar.gz>`. Tar-filen innehåller både källkod, kompilerade körbara filer och manualblad.

För att se till att bara auktoriserade användare skall kunna köra *sliplogin*, så bör man lägga till en rad i filen `/etc/group`, ungefär som följande:

```
 ..
slip::13:radio,fred
 ..
```

När man installerar *sliplogin*-paketet så kommer `Makefile` att ändra ägargruppen för kommandot *sliplogin* till `slip`, och det betyder att endast användare som tillhör den gruppen kan köra programmet. I exemplet ovan kan endast `radio` och `fred` köra *sliplogin*.

För att installera binärfilerna i katalogen `/sbin` gör man följande:

```
# cd /usr/src
# gzip -dc .../sliplogin-2.1.1.tar.gz | tar xvf -
# cd sliplogin-2.1.1
# <..edit the Makefile if you don't use shadow passwords..>
# make install
```

Om man vill kompilera om binärfilerna innan man installerar så skall man göra `make clean` innan man kör `make install`. Om man vill installera filerna i någon annan katalog så får man ändra i filen `Makefile` under regeln *install*.

Läs filen `README` som följer med paketet för mer information.

Att konfigurera `/etc/passwd` för SLIP. Normalt skulle man skapa särskilda login för SLIP-användare i filen `/etc/passwd`. En konvention som ofta följs är att använda *hostname* för den anslutande datorn med ett stort 'S' i början. Så, till exempel, om den anslutande datorn heter `radio` så kan man lägga en rad i `/etc/passwd` som ser ut såhär:

```
Sradio:FvKurok73:1427:1:radio SLIP login:/tmp:/sbin/sliplogin
```

Men det spelar egentligen ingen roll vad kontot heter, bara det betyder något för administratören av SLIP-servern.

Notera att användarna inte behöver en särskild hemkatalog, efter som de inte kommer att få något shell från denna maskinen. Därför duger `/tmp` gott. Notera även att *sliplogin* används istället för ett normalt shell.

Att konfigurera `/etc/slip.hosts` Filen `/etc/slip.hosts` är filen som *sliplogin* söker igenom efter rader som stämmer överens med loginnamnet för att få tag på konfigurationsdetaljer för denna användare. Det är filen där man specificerar IP-adressen och nätmasken som kommer att tilldelas användaren. Exempelrader för två datorer, en statisk konfiguration för `radio` och en annan, dynamisk konfiguration för `albert` kan se ut såhär:

```
#
Sradio   44.136.8.99   44.136.8.100  255.255.255.0  normal      -1
Salbert  44.136.8.99   DYNAMIC       255.255.255.0  compressed  60
#
```

Parametrarna i `/etc/slip.hosts` är:

1. loginnamnet för användaren.
2. IP-adress för servermaskinen (dvs denna dator)
3. IP-adress som användaren blir tilldelad. Om detta fält innehåller `DYNAMIC` så kommer IP-adressen att allokeras baserad på informationen som finns i filen `/etc/slip.tty`. **Notera:** man måste ha åtminstone version 1.3 av *sliplogin* för att detta skall fungera.
4. nätmasken som tilldelas den anslutande maskinen.
5. SLIP-läge där man kan sätta på/stänga av kompression och andra funktioner. Tillåtna värden är `normal` eller `compressed`.
6. en timeoutparameter som specificerar hur lång tid länken kan vara oanvänd innan den automatiskt avbryts. Ett negativt värde stänger av funktionen.
7. valfria argument.

Notera att man kan använda antingen datornamn eller IP-adresser för fält 2 och 3. Om man använder datornamn så måste dessa kunna översättas, dvs maskinen måste måste kunna hitta en IP-adress som stämmer överens med datornamnet annars kommer scripet att fallera när det körs. Man kan testa detta genom att försöka öppna en telnet-anslutning till datornamnet, om man får meddelandet '*Trying nnn.nnn.nnn...*' så kan maskinen hitta en IP-adress. Får man däremot meddelandet '*Unknown host*' så kan den inte hitta någon IP-adress. Om inte så använder man IP-adresser eller fixar till konfigurationen av sin namnöversättare (se avsnitt 'Att konfigurera din Name Resolver').

De vanligaste SLIP-lägena är:

normal

att slå på normal SLIP utan kompression.

compression

att slå på van Jacobsen header compression (cSLIP)

Man kan endast använda en åt gången. För mer information om de andra valmöjligheterna som finns, se manualbladen.

Att konfigurera `/etc/slip.login`. När *sliplogin* har hittat en passande rad i `/etc/slip.hosts` så kommer programmet att försöka exekvera filen `/etc/slip.login` för att konfigurera SLIP-gränssnittet med dess IP-adress och nätmask.

Exemplet på en `/etc/slip.login` som följer med *sliplogin*-paketet ser ut så här:

```
#!/bin/sh -
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90
#
# generic login file for a SLIP line.  sliplogin invokes this with
# the parameters:
#     $1       $2       $3    $4, $5, $6 ...
#   SLIPunit ttyspeed   pid   the arguments from the slip.host entry
#
/sbin/ifconfig $1 $5 pointopoint $6 mtu 1500 -trailers up
/sbin/route add $6
arp -s $6 <hw_addr> pub
exit 0
#
```

Scriptet använder helt enkelt kommandona *ifconfig* och *route* för att konfigurera SLIP-enheten med dess IP-adress, avlägsen IP-adress och nätmask och sedan skapa en route till den avlägsna adressen via SLIP-enheten. Precis som om man använde kommandot *slattach*.

Notera också användandet av *Proxy ARP* för att se till att andra datorer på samma Ethernet som servern kommer att nå den anslutna datorn. Fältet <`hw_addr`> skall vara hårdvaruadressen på Ethernet-kortet i maskinen. Om servern inte är ansluten till ett Ethernet-nätverk kan man helt utelämna denna rad.

Att konfigurera `/etc/slip.logout`. När anslutningen avslutas, så vill man se till att den seriella enheten återställs till sitt normala tillstånd så att framtida anslutningar kan logga in ordentligt. Detta uppnås genom med hjälp av filen `/etc/slip.logout`. Dess format är ganska enkelt och används med samma argument som filen `/etc/slip.login`.

```
#!/bin/sh -
#
#               slip.logout
#
/sbin/ifconfig $1 down
arp -d $6
exit 0
#
```

Allt den gör är 'ta ner' gränssnittet vilket kommer att ta bort routen som skapades tidigare. Den använder också kommandot *arp* för att ta bort eventuella *Proxy ARP*s som har skapats, återigen, man behöver inte kommandot *arp* ifall servern inte är ansluten till ett Ethernet-nätverk.

Att konfigurera `/etc/slip.tty`. Om man använder dynamisk IP-adressallokering (har någon rad konfigurerad med `DYNAMIC` i `/etc/slip.hosts`) så måste man konfigurera filen `/etc/slip.tty` för att lista vilka adresser som tilldelas vilken port. Man behöver endast denna fil om man vill att servern dynamiskt skall ge adresser till användare. Formatet är följande:

```
# slip.tty    tty -> IP address mappings for dynamic SLIP
```

```
# format: /dev/tty?? xxx.xxx.xxx.xxx
#
/dev/ttyS0      192.168.0.100
/dev/ttyS1      192.168.0.101
#
```

Vad denna fil säger är att användare som ansluter till porten `/dev/ttyS0` och som har sitt adressfält i filen `/etc/slip.hosts` satt till `DYNAMIC` kommer att tilldelas adressen `192.168.0.100`.

På detta sättet behöver man endast allokera en adress per port förutom för de användare som behöver en egen adress. Detta hjälper till att hålla nere antalet adresser man behöver till ett minimum och undviker slösande.

6.28.2 Slip Server med *dip*.

Låt mig börja med att säga att lite av informationen nedan kom ifrån manualbladet för *dip*, där det kortfattat beskrivs hur man kör Linux som en SLIP-server. Var också uppmärksam på att det som följer är baserat på paketet *dip337o-uri.tgz* och antagligen inte gäller för andra versioner av *dip*.

dip har ett läge när det automatiskt letar upp en rad för den användaren som körde programmet och konfigurerar den seriella linjen som en SLIP-länk baserat på informationen som den hittar i filen `/etc/diphosts`. Detta läge aktiveras genom att köra *dip* som *diplogin*. Detta är därför sättet på vilket man använder *dip* som en SLIP-server, genom att skapa särskilda konton där *diplogin* används som login-shell. Det första som skall göras är att göra en symbolisk länk som följer:

```
# ln -sf /usr/sbin/dip /usr/sbin/diplogin
```

Sedan behöver man lägga till rader i filerna `/etc/passwd` och `/etc/diphosts`. Raderna är formaterade som följer:

För att konfigurera Linux som en SLIP-server med *dip* behöver man skapa några särskilda SLIP-konton för användare, där *dip* används som login-shell. En föreslagen konvention är att låta alla SLIP-konton börja med ett stort 'S', tex 'Sfredm'.

Ett exempel på en rad för en SLIP-användare i `/etc/passwd` ser ut så här:

```
Sfredm:ij/SMxiTlGVCo:1004:10:Fred:/tmp:/usr/sbin/diplogin
^^      ^^          ^^  ^^   ^^   ^^   ^^
|       |           |   |    |    |    \__ diplogin as login shell
|       |           |   |    |    \_______ Home directory
|       |           |   |    \____________ User Full Name
|       |           |   \_________________ User Group ID
|       |           \_____________________ User ID
|       \_________________________________ Encrypted User Password
\__________________________________________ Slip User Login Name
```

När användaren har loggat in så utför programmet *login*, om det hittar och verifierar användaren ok, kommandot *diplogin*. *dip*, när det körs som *diplogin*, vet automatiskt att det skall användas som ett login-shell. När det startas som *diplogin* så är det första programmet gör att använda funktionsanropet *getuid()* för att få användarid för den som körde programmet. Det letar sedan i filen `/etc/diphosts` efter den första raden som passar in på det användaridt eller namnet på den *tty*-enhet som anslutningen kom från och konfigurerar sig själv därefter. Genom att välja om en användare skall få en rad i filen `/etc/diphosts` eller om användaren skall ges den generella konfigurationen så kan man bygga sin server på så sätt att man kan ha en blandning av statiskt och dynamiskt tilldelade adresser för användarna. *dip* kommer automatiskt att lägga till en *Proxy ARP* så detta behöver man inte bry sig om att göra manuellt.

Att konfigurera `/etc/diphosts` Filen `/etc/diphosts` används av *dip* för att hitta konfigurationer för avlägsna datorer. Dessa avlägsna datorer kan vara användare som ringer in till en Linuxbox eller de kan vara datorer som man ringer till ifrån Linuxboxen.

Det generella formatet på `/etc/diphosts` är som följer:

```
 ..
Suwalt::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
ttyS1::145.71.34.3:145.71.34.2:255.255.255.0:Dynamic ttyS1:CSLIP,296
 ..
```

Fälten är:

1. `login name`: som returnerat av getpwuid(getuid()) eller tty namn.
2. `oanvänd`: kompatibilitet med /etc/passwd
3. `Remote Address`: IP-adress för den anslutande datorn, antingen numeriskt eller med namn
4. `Local Address`: IP-adress för denna dator, numeriskt eller med namn
5. `Netmask`: i punkterad decimal notation
6. `Comment field`: skriv vad du vill här.
7. `protocol`: Slip, CSlip osv.
8. `MTU`: decimalt tal

Ett exempel på en rad i `/etc/net/diphosts` för en SLIP-användare kan vara:

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:SLIP,296
```

som specificerar en SLIP-länk med en avlägsen adress 145.71.34.2 och MTU på 296, eller:

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
```

som specificerar en cSLIP-länk med avlägsen adress 145.71.34.1 och MTU på 1006.

Därför så skall alla användare som man vill tillåta en statiskt allokerad adress ha en rad i `/etc/diphosts`. Om man vill att användare som ansluter till en viss port skall få en dynamisk allokerad adress så måste man ha en rad för `tty`-enheten och ingen rad för användaren. Man skall komma ihåg att konfigurera åtminstonde en rad för varje `tty`-enhet som används för SLIP så att det säkert finns en passande konfiguration oavsett vilket modem de ringer in på.

När användare loggar in får de en normal login och password prompt vid vilken de skall skriva in sitt SLIP-login och password. Om dessa verifieras ok så kommer användaren inte se några särskilda meddelanden och användarna skall endast byta till SLIP-läge på sin sida. Användaren bör ha möjlighet att ansluta ok och bli konfigurerad med relevanta parametrar från filen `diphosts`.

6.28.3 SLIP server med paketet *dSLIP*

Matt Dillon <`dillon@apollo.west.oic.com`> har skrivit ett programpaket som inte bara ringer in utan också ringer ut med SLIP. Matts paket är en kombination av små program och scripts som hanterar ens anslutningar. För detta måste man ha *tcsh* eftersom åtminstone ett av scripten kräver detta. Matt tillhandahåller en binär kopia av verktyget *expect* eftersom även det behövs av en del av scripten. Man behöver antagligen ha lite erfarenhet med kommandot *expect* för att lyckas länka in ok, men låt inte detta avskräcka.

Matt har skrivit bra med installationsinstruktioner i README-filen så jag tänker inte upprepa dem.

Man kan hämta *dSLIP* från dess hemsida på:

apollo.west.oic.com

```
/pub/linux/dillon_src/dSLIP203.tgz
```

eller från:

sunsite.unc.edu

```
/pub/Linux/system/Network/serial/dSLIP203.tgz
```

Läs filen `README` och skapa raderna i `/etc/passwd` och `/etc/group` **innan** `make install`.

6.29 STRIP (Starmode Radio IP)

Kompileringsalternativ för Kärnan:

```
Network device support  --->
        [*] Network device support
        ....
        [*] Radio network interfaces
        < > STRIP (Metricom starmode radio IP)
```

STRIP är ett protokoll som designats speciellt för en typ av Metricom radiomodem för ett forskningsprojekt på Stanford University som heter *MosquitoNet Project* `<http://mosquitonet.Stanford.EDU/mosquitonet.html>`. Det finns mycket intressant att läsa där, även om man inte är direkt intresserad av projektet.

Metricom radiorna kopplas till en seriell port, använder många olika tekniker och klarar av en överföringshastighet på ungefär 100kbps. Information om Metricom radior kan fås på: *Metricom Web Server* `<http://www.metricom.com/>`.

För närvarande så stöds inte STRIP av standardverktygen för nätverk, så man måste ladda hem några specialdesignade verktyg från MosquitoNets WWW-server. Detaljer om vad man behöver finns på: *MosquitoNet STRIP Page* `<http://mosquitonet.Stanford.EDU/strip.html>`.

En sammanfattning av konfigurationen är att man använder ett modifierat *slattach* program för att ställa in den seriella enheten i STRIP-läge och sedan konfigurera den resulterande '`st[0-9]`'-enheten som om det var en Ethernet-enhet, men med en viktig skillnad. Av tekniska skäl så stöder inte STRIP ARP-protokollet, så man måste konfigurera ARP-raderna för var och en av datorerna på sitt subnät manuellt.

6.30 Token Ring

Enhetsnamn för Token Ring är '`tr0`', '`tr1`' osv. Token Ring är ett standardiserat LAN-protokoll från IBM som undviker kollisioner med en mekanism som endast tillåter en station på LANet att sända åt gången. En 'token' innehas av en station åt gången och stationen som har 'token' får lov att sända. När den har sänt sin data så skickar den 'token' vidare till nästa station. Token går runt mellan alla aktiva stationer, därav namnet 'Token Ring'.

Kompileringsalternativ för Kärnan:

```
Network device support  --->
        [*] Network device support
        ....
        [*] Token Ring driver support
        < > IBM Tropic chipset based adaptor support
```

Konfigurationen av Token Ring är identisk med den som görs med Ethernet förutom att man använder andra enhetsnamn.

6.31 X.25

X.25 är ett paketswitchat protokoll som definieras av `C.C.I.T.T.` (en standardiseringsorganisation som erkänns av de flesta telebolag i världen). En implementation av X.25 och LAPB håller på att utvecklas och de senaste `2.1.*` kärnor inkluderar detta arbete.

Jonathon Naylor `jsn@cs.nott.ac.uk` leder utvecklingen och det har skapats en mailinglista för att diskutera ämnen relaterade till Linux X.25. För att prenumerera skall man skicka ett meddelande till `majordomo@vger.rutgers.edu` med texten ”`subscribe linux-x25`” i meddelandekroppen.

Tidiga versioner av konfigureringsverktygen kan hämtas från Jonathons ftp-sajt på *ftp.cs.nott.ac.uk* `<ftp://ftp.cs.nott.ac.uk/jsn/>`.

6.32 WaveLan

Enhetsnamn för WaveLan är ‘`eth0`’, ‘`eth1`’, osv.

Kompileringsalternativ för Kärnan:

```
Network device support  --->
        [*] Network device support
        ....
        [*] Radio network interfaces
        ....
        <*> WaveLAN support
```

WaveLan-kortet är ett vitt spektrum av trådlösa LAN-kort. Korten liknar Ethernet-kort mycket och konfigureras på ungefär samma sätt.

Mer information om WaveLan finns på *Wavelan.com* `<http://www.wavelan.com/>`.

7 Kablage

De som är händiga med en lödpenna kanske vill bygga egna kablar för att koppla ihop Linuxburkar. Följande figurer bör vara till hjälp för detta.

7.1 Seriell NULL-modem kabel.

Alla NULL-modem kablar är **inte** likadana. Många NULL-modem kablar gör inte mer än att lura datorn att tro att alla signaler finns och byter sänd och ta emot data pinnarna. Detta fungerar men betyder att man måste använda flödeskontroll i mjukvara (XON/XOFF) vilket inte är så effektivt som flödeskontroll i hårdvara. Följande kabel har bästa möjliga signalering mellan datorer och tillåter användning av flödeskontroll i hårdvara (RTS/CTS).

```
Pin Name  Pin                               Pin
Tx Data    2  ----------------------------  3
Rx Data    3  ----------------------------  2
RTS        4  ----------------------------  5
CTS        5  ----------------------------  4
Ground     7  ----------------------------  7
DTR        20 -\--------------------------  8
DSR        6  -/
RLSD/DCD   8  --------------------------/-  20
                                        \-  6
```

7.2 Parallellportskabel (PLIP kabel)

Om man tänker använda PLIP-protokollet mellan två maskiner så fungerar följande kabel oavsett vilken typ av parallellportar man har.

```
Pin Name    pin         pin
STROBE      1*
D0->ERROR   2  ----------- 15
D1->SLCT    3  ----------- 13
D2->PAPOUT  4  ----------- 12
D3->ACK     5  ----------- 10
D4->BUSY    6  ----------- 11
D5          7*
D6          8*
D7          9*
ACK->D3     10 ----------- 5
BUSY->D4    11 ----------- 6
PAPOUT->D2  12 ----------- 4
SLCT->D1    13 ----------- 3
FEED        14*
ERROR->D0   15 ----------- 2
INIT        16*
SLCTIN      17*
GROUND      25 ----------- 25
```

Anmärkningar:

- Anslut inte pinnarna som är markerade med en asterisk '*'.
- Extra jordning är 18,19,20,21,22,23 och 24.
- Om kabeln man använder har en metallskärmning, så skall den anslutas till metallskalet på DB-25 **endast på en sida**.

Varning: en felkopplad PLIP-kabel kan förstöra kontrollerkortet. Var försiktig och dubbelkolla alla anslutningar.

Även om man kan använda PLIP-kablar över långa avstånd så skall man undvika det om man kan. Specifikationerna för kabeln tillåter längder på ungefär en meter. Var försiktig med långa PLIP-kablar eftersom källor för stark elektromagnetisk strålning (tex blixtar, elkablar och radiosändare) kan störa och till och med skada kontrollerkortet. Om man verkligen vill ansluta två datorer över långa avstånd så bör man titta på möjligheterna med Ethernet och använda tunn koaxialkabel.

7.3 10base2 (tunn koax) Ethernet-kabel

10base2 är en kabelstandard för Ethernet som specificerar en 52 ohms koaxialkabel med en diameter på ungefär 5 millimeter. Det finns ett par viktiga regler att komma ihåg när man skall koppla ihop maskiner med 10base2 kablar. Det första är att man måste använda terminatorer i **båda ändar** av kabeln. En terminator är ett 52 ohms motstånd som ser till att signalen absorberas, och inte reflekteras, när den når slutet av kabeln. Utan en terminator i båda ändar av kabeln så kommer man att märka att nätverket är opålitligt eller inte fungerar alls. Normalt så använder man 'T-korsningar' för att koppla ihop maskinerna, så man får något som liknar:

```
|==========T=============T=============T==========T==========|
           |             |             |          |
           |             |             |          |
         -----         -----         -----      -----
         |   |         |   |         |   |      |   |
         -----         -----         -----      -----
```

där en '|' i båda ändar representerar en terminator, '======' representerar en längd med koaxialkabel med BNC-kontakter i båda ändar och 'T' representerar en 'T-korsning'-kontakt. Man skall försöka att hålla längden på kabeln mellan 'T-korsningen' och Ethernet-kortet så kort som möjligt, det bästa är om 'T-korsningen' är kopplad direkt på Ethernet-kortet.

7.4 Tvinnad tvåpar Ethernet-kabel.

Om man bara har två Ethernet-kort med tvinnad tvåpar anslutning som man vill koppla ihop så måste man inte ha en hub. Man kan koppla ihop korten direkt. Ett diagram som visar hur man gör detta finns i *Ethernet-HOWTO* `<Ethernet-HOWTO.html>`.

8 Terminologi i detta dokument.

Följande är en lista med de viktigaste termerna som används i detta dokument.

ARP

Detta är en förkortning av *Address Resolution Protocol* och det är hur en maskin på ett nätverk associerar en IP-adress med en hårdvaruadress.

ATM

Detta är en förkortning av *Asynchronous Transfer Mode*. Ett ATM-nätverk packar data i block med en standardstorlek vilka det sedan kan skicka effektivt från punkt till punkt. ATM är ett kopplingsorienterat nätverk med virtuella kretsar.

klient

Detta är vanligtvis mjukvaran i den ändan av systemet där användaren finns. Det finns undantag, till exempel i X11 fönstersystem så är det egentligen servern som finns hos användaren och klienten kör på en annan maskin. Klienten är programmet eller delen av systemet som använder en tjänst som tillhandahålls av servern. I fallet med *peer to peer* system, som *slip* eller *ppp*, så är den dator som startar anslutningen klienten och den andra datorn servern.

datagram

Ett datagram är ett diskret paket med data och huvuden, som innehåller adresser, som är en transmissionsenhet över ett IP-nätverk. Detta kan också kallas för 'paket' i vissa sammanhang.

DLCI

Detta är en förkortning av *Data Link Connection Identifier* och används för att identifiera en unik virtuell punkt till punkt anslutning via ett Frame Relay nätverk. DLCIs delas normalt ut av den som tillhandahåller Frame Relay nätverket.

Frame Relay

Frame Relay är en nätverksteknik som är anpassad för att bära trafik som är av oregelbunden natur. Nätverkskostnader reduceras genom att flera Frame Relay kunder delar samma nätverkskapacitet och förväntas använda kapaciteten vid olika tidpunkter.

hårdvaruadress

Detta är ett nummer som unikt identifierar en dator i ett nätverk på det länklagret. Exempel på detta är *Ethernetadresser* och *AX.25-adresser*.

ISDN

Detta är en förkortning av *Integrated Services Digital Network*. ISDN tillhandahåller ett standardiserat sätt på vilket telebolag kan erbjuda antingen data- eller rösttjänster till en kund. Tekniskt sett är ISDN ett datanätverk med virtuella kretsar.

ISP

Detta är en förkortning av *Internet Service Provider* (internetleverantör). Detta är de organisationer eller företag som erbjuder nätverksanslutningar till Internet.

IP-adress

Detta är ett nummer som unikt identifierar en TCP/IP dator på ett nätverk. Adressen är 4 bytes lång och representeras ofta i vad som heter "*punkterad decimal notation*", där varje byte skrivs för sig och de olika bytearna är separerade med en '.'.

MTU

Detta är en förkortning av *Maximum Transmission Unit*. MTU är en parameter som bestämmer den största datagramstorleken som kan överföras av ett IP-gränssnitt utan att det behöver delas upp i flera mindre delar. MTU bör vara större än det största datagram som man vill skicka ofragmenterat. Notera att detta bara hindrar lokal fragmentering, om datagrammet passerar en länk med en mindre MTU så kan det fragmenteras där. Typiska värden för MTU är 1500 bytes för Ethernet och 576 bytes för SLIP.

route

Routen (eller rutten) är den väg som datagrammen färdas genom nätverket för att nå sin destination.

server

Detta är vanligtvis den mjukvara eller del av systemet som är avlägset från användaren. Servern erbjuder tjänster till en eller flera klienter. Exempel på servrar är *ftp*, *Networked File System* och *Domain Name Server*. I fallet med *peer to peer* system, som *slip* eller *ppp*, så är den dator som startar anslutningen klienten och den andra datorn servern.

9 Linux för en ISP?

Om man är intresserad av att använda Linux för ISP-syften så rekommenderar jag att titta på *Linux ISP homepage* `<http://www.anime.net/linuxisp/>` för en bra lista med pekare till information som kan vara användbar.

10 Tillkännagivanden

Jag skulle vilja tacka följande personer för bidrag till detta dokument (ingen speciell ordningsföljd): Terry Dawson, Axel Boldt, Arnt Gulbrandsen, Gary Allpike, Cees de Groot, Alan Cox, Jonathon Naylor, Claes Ensson, Ron Nessim, John Minack, Jean-Pierre Cocatrix, Erez Strauss.

11 Copyright.